IDENTIFICACIÓN Y REMOCIÓN DE PIEZAS PARA EL ENSAMBLE DE BOTONES MEDIANTE VISIÓN ARTIFICIAL

MIGUEL VELÁSQUEZ CAMPILLO

Trabajo de grado para optar al título de Ingeniero Mecatrónico

Víctor Hugo Jaramillo Velásquez

UNIVERSIDAD EIA
INGENIERÍA MECATRÓNICA
ENVIGADO
2018

CONTENIDO

pág.

LISTA DE TABLAS

LISTA DE ILUSTRACIONES

LISTA DE CÓDIGOS

RESUMEN

En el presente trabajo se diseña y desarrolla un sistema de identificación y remoción de piezas para el ensamble de botones metálicos, mediante técnicas de visión artificial y *Machine Learning*. El sistema consta principalmente de una parte para la adquisición de imágenes en la línea de ensamble, y otra para la remoción de dichas piezas cuando se encuentran al revés. Se comienza con el diseño experimental físico de estas partes: la base para la cámara que detecta las piezas, y el sistema para remover las piezas directamente en el riel de ensamble. Luego se procede a desarrollar los distintos algoritmos en Python, utilizando librerías de OpenCV, para analizar e identificar si una pieza en particular se encuentra al revés, y en caso de serlo, expulsarla con el sistema de remoción. La técnica de entrenamiento del sistema de identificación es utilizando características geometrías para determinar las diferencias entre las piezas. Finalmente se instala el dispositivo de selección en una maquina ensambladora y se procede a verificar su funcionamiento, evaluando la capacidad de identificar y remover correctamente las piezas, al igual que evaluar su porcentaje de confiabilidad al realizar la tarea. El trabajo fue realizado para 3 referencias de botones, en donde se obtuvieron porcentajes de identificación y remoción de 90.4% para la referencia 2 y 91.6% para la referencia 3. La referencia 1 resultó ser demasiado similar al derecho y al revés como para poder determinar con certeza y con factores geométricos su diferencia.

Palabras clave: Visión Artificial, Machine Learning, Python, OpenCV

ABSTRACT

In the present document, a system of identification and removal of pieces for the assembly of metallic buttons is designed and developed, by means of artificial vision techniques and Machine Learning. The system consists mainly of one part for the acquisition of images in the assembly line, and another for the removal of said pieces when they are in reverse. It begins with the experimental physical design of two parts: the base for the camera that detects the pieces, and the system for removing the pieces directly on the assembly rail. Next, we proceed with the development of different algorithms in Python, using OpenCV libraries, to analyze and identify if a particular piece is upside down, and if it is, it should be ejected by the removal system. The training technique of the identification algorithms is mainly focused on using geometrical characteristics to determine the differences between the pieces in their different orientations. Finally, the selection device is installed in an assembly machine and its operation is verified, evaluating the ability to identify and remove the pieces correctly, as well as evaluating their percentage of reliability when performing the task. The work was done for 3 references of buttons, where percentages of identification and removal of 90.4% were obtained for reference 2 and 91.6% for reference 3. Reference 1 turned out to be too similar on both faces of to be able to determine with certainty and with geometric characteristics their difference.

Keywords: Artificial Vision, Machine Learning, Python, OpenCV

INTRODUCCIÓN

En el ensamble automático de botones metálicos, las piezas que componen al botón viajan por rieles para organizarse unas encima de otras. Las piezas suben por un tambor vibrador, que organiza las piezas en un riel donde se organizan en una fila para continuar el proceso de ensamble. El proceso necesita la supervisión humana de un operario, para asegurarse de voltear las piezas que vienen al revés desde el riel. En el presente trabajo se diseña y desarrolla un sistema de identificación y remoción de piezas para el ensamble de botones metálicos, mediante técnicas de visión artificial y *Machine Learning*. El sistema consta principalmente de una parte para la adquisición de imágenes en la línea de ensamble, y otra para la remoción de dichas piezas cuando se encuentran al revés. Se comienza con el diseño físico de estas partes: la base para la cámara que detecta las piezas, y el sistema para remover las piezas directamente en el riel de ensamble. Luego se procede a desarrollar los distintos algoritmos en Python para analizar e identificar si una pieza en particular se encuentra al revés, y en caso de serlo, expulsarla con el sistema de remoción. Finalmente se instala el dispositivo de selección en una maquina ensambladora y se verifica su funcionamiento.

1 PRELIMINARES

1.1 PLANTEAMIENTO DEL PROBLEMA

En el mundo actual, la tecnología ha avanzado hasta el punto donde ciertos procesos industriales pueden ser desarrollados con más facilidad, velocidad y precisión en comparación a años anteriores, gracias a las tecnologías en control, automatización industrial y visión artificial. Una de estas actividades en específico, en el área industrial es el ensamble de productos.

Actualmente la separación de las piezas de botones metálicos de golpe en la línea de ensamble automático es hecha totalmente de forma mecánica, mediante geometrías específicas para cada pieza. La separación se hace en tambores vibradores, donde las estrías y canales llevan las piezas de un bulto desordenado a una fila ordenada de piezas. La vibración permite el movimiento de las piezas dentro del tambor y los obstáculos por los que pasan orientan las piezas de una forma deseada para continuar con el ensamble. Por ser un proceso que solo depende de la geometría del tambor vibrador, y sus solo es posible separar un tipo de piezas con unas dimensiones y peso determinado.

Como ejemplo, en la empresa C.I. Estrada Velasquez se encuentran distintos tambores para cada estilo de pieza, las cuales solo operan con un producto, y para trabajar con una pieza de dimensiones distintas habría que cambiar el tambor entero. ¿Qué tipo de sistema automático puede ser implementado en un tambor vibrador para separar las piezas que se encuentren un una orientación errónea para el proceso de ensamble?

1.2 OBJETIVOS DEL PROYECTO

1.2.1 Objetivo General

Desarrollar un sistema automático de identificación y separación de piezas desordenadas utilizando visión artificial en la línea de ensamble.

1.2.2 Objetivos Específicos

1.2.2.1 Diseñar un dispositivo de identificación y remoción que cuente con un sistema de detección por visión artificial y un separador de piezas de la línea de ensamble.

1.2.2.2 Desarrollar un algoritmo para el sistema de detección por visión artificial.

1.2.2.3 Implementar el dispositivo funcional en un tambor vibrador.

1.2.2.4 Verificar el funcionamiento del dispositivo en la empresa C. I. Estrada Velásquez

1.3 MARCO DE REFERENCIA

1.3.1 Antecedentes

Hoy en día, la mejora en la tecnología de comunicación, cámaras más rápidas y asequibles, servomotores y dispositivos de control más baratos y más pequeños, han permitido el diseño de un servo mecanismo simple y de precio razonable (García, F. & Morales, A., 2016). Los autores utilizaron la nueva tecnología para realizar un sistema robótico para la identificación la ubicación euclidiana de objetos rojos simulando tomates por medio de visión artificial. En el proyecto se construye un mecanismo para alcanzar la posición donde se encuentra el objeto rojo después de que se ha visualizado y procesado la imagen tomada utilizando servomotores y una cámara ASUS XTion Pro Live. La utilización de estos instrumentos y el uso de OpenCV para el procesamiento de imágenes son de gran relevancia para la construcción de una empacadora automática al revelar posibles partes para utilizar en el proyecto (García, F. & Morales, A., 2016).

(Porras, De la Cruz, & Morán, 2014) diseñaron e implementaron un sistema automático para el proceso de control de calidad y clasificación de objetos. Fue necesario dividir el proyecto en dos partes. La primera consistía en el desarrollo y construcción del hardware; el cual tenía una banda transportadora en la cual se movilizaban los objetos que iban a ser procesados, además contaba con la cámara, el sistema de iluminación, sensores de luz, circuitos integrados y motores paso a paso; para que se pudiera realizar el proceso de obtención y análisis de la imagen. Por otra parte, el software se encargaba de realizar todo el proceso de caracterización de los objetos. Los resultados del proyecto fueron satisfactorios y se concluyó que al utilizar herramientas como la visión artificial en la automatización de procesos, se pueden reducir costos y mejorar la eficiencia de las tareas.

También, (Malpartida & Sotelo, s. f.) utilizan algoritmos para el reconocimiento de piezas basados en visión artificial. El objetivo del proyecto fue de implementar un sistema robótico para el reconocimiento y localización de piezas. Primero utilizaron un sistema de iluminación para poder captar las imágenes de forma adecuada, después la imagen es procesada para eliminar el ruido y mejorarla. Posteriormente el sistema segmenta la imagen y le extrae sus características para que se pueda reconocer el objeto. Finalmente el sistema se encarga de controlar el robot y que este sea capaz de ubicar las piezas en la posición deseada. Los autores concluyeron que estos sistemas robóticos basados en visión artificial, permiten la optimización en las industrias.

En la clasificación automática industrial también se utilizan elementos de instrumentación como los sensores, como lo realizado por (Zurita & Pérez, 2014), los cuales diseñaron y construyeron una máquina clasificadora de objetos basados en la detección de un sensor de color. El sistema se basa en un software que procesa las señales enviadas por los sensores y obtenidas por una tarjeta de adquisición de datos. Este determina la acción

que debe realizar un brazo robótico que es el encargado de clasificar los objetos. Después de la implementación del sistema, se obtuvieron resultados exitosos y el prototipo fue capaz de identificar los objetos según su color y ubicarlos en los lugares determinados.

(Lozano Nasner & Bayona Ibañez, 2003) diseñaron e implementaron un sistema para la clasificación de objetos en un proceso industrial, basados en técnicas de visión artificial. Al sistema se le indica el objeto patrón que debe reconocer. Los objetos son llevados por una banda transportadora y un brazo robótico es el encargado de clasificarlos como aceptados o rechazados. El brazo robótico toma los objetos que son rechazados y los ubica en una posición predeterminada. Los autores obtuvieron resultados exitosos donde el sistema funcionaba correctamente según lo diseñado inicialmente.

En los cultivos de café, (Sandoval Niño & Prieto Ortiz, 2007) identificaron un problema para determinar el momento de recolección de los frutos. Para esto, desarrollaron un sistema de visión artificial para la clasificación del café según su estado de madurez, el cual se basa en la forma y el color del fruto. Para la descripción de la forma y el color se extrajeron 208 características de las imágenes y finalmente las reducen a 9. Para la clasificación utilizaron dos técnicas: clasificador Bayesiano y redes neuronales. Los resultados arrojaron un error de clasificación de 5,43% y un tiempo de 5,5 ms para el clasificador Bayesiano, mientras que la red neuronal 7,46% y 0,8 ms respectivamente.

En la industria farmacéutica también se han desarrollado proyectos de clasificación automático como el de (Saldaña González, Estévez Carrerón, & Silva Ortigoza, 2014). En este, se implementó un sistema que permitiera clasificar los productos a ser almacenados según su color, con el objetivo de mantener un control de las existencias para mejorar la gestión de suministros. El sistema de clasificación y control se desarrolló en LabView, el cual se comunica con un PIC18F255 que se conecta con una serie de sensores y actuadores sobre una banda transportadora que es la encargada de ubicar las cajas de productos según su color.

La visión artificial ha demostrado ser de gran eficiencia en sistemas de separación y selección en el área de producción y control de calidad. (Sofu, Er, Kayacan, & Cetisli, 2016) crearon un sistema para organizar manzanas de distintos tipos mediante visión artificial. Construyen una cabina donde se van a analizar las manzanas con iluminación adecuada, y una locomoción de la banda transportadora. Los factores que se les miden a las manzanas son el tamaño, peso, zonas defectuosas y color. Se deseaba hacer una máquina separadora de manzanas que separa aproximadamente 200,000 manzanas al día con más de 75% de eficacia, y se logró construir y programar una máquina capaz de organizar y separar aproximadamente 432,000 manzanas al día con una exactitud de 79%, por lo que es correcto afirmar que el resultado fue mejor del esperado.

Wilfred M. Bourg, entre otros, es el inventor de un método para extraer información de las características de las imágenes de producto mediante el análisis de imágenes multi-variable basado en el Análisis de Componentes Principales (PCA). Se utiliza para desarrollar modelos predictivos para las características y la distribución del producto fotografiado. El sistema de imágenes se utiliza para controlar las variables de la calidad

del producto en un entorno de fabricación en línea. También puede ser integrado en el sistema de control de retroalimentación de bucle cerrado en sistemas automatizados (Estados Unidos Patente nº US70688117B2, 2006).

En el área industrial es también visto el control de calidad con visión artificial. En 2016, Young-Jin Cha, Kisung You y Wooram Choi plantearon una forma para detectar pernos sueltos que necesiten ser ajustados. A partir de las imágenes tomadas, las características simples sensibles al daño, tales como las longitudes horizontal y vertical de la cabeza del perno se calcularán automáticamente utilizando la transformación de Hough y otras técnicas de procesamiento de imágenes. A pesar de las muchas ventajas del algoritmo propuesto, quedan algunas limitaciones, y son necesarias otras modificaciones del algoritmo para diferentes diseños y diferentes tipos de pernos. Además, en esta etapa de desarrollo, el algoritmo sólo puede detectar pernos con distorsión distintiva con ángulos y distancias de cámara limitados, tales como ángulos que van desde 31 ° a 51 ° y distancias que van desde 78 mm hasta 122 mm (Young-Jin, Kisung, & Wooram, 2016).

JasVisio, una empresa que desarrolla sistemas de visión artificial para el control de calidad, el posicionado de piezas, reconocimiento, inspección y detección de fallos sin contacto opera en distintas zonas de la industria. La aplicación en áreas como las industrias de alimentación, industria automotora, electrónica, industria farmacéutica, packaging entre otras es de las principales en el área de la empresa. Comúnmente, la importancia de las inspecciones realizadas es el monitoreo continuo y automático (JasVisio, 2009).

1.3.2 Marco Teórico

El ensamble de botones metálicos de golpe y remaches en el área de la marroquinería consiste en unir mediante embutido o troquelado dos partes del botón. Estas partes son generalmente de latón, hierro, zamac o latón, las cuales vienen de un proceso de embutido donde se les da la forma necesaria para ser ensambladas y los logotipos de las marcas que se van a ver en los botones. Las partes denominadas como cuello y tapa deben estar una encima de la otra y en una orientación determinada para formar un botón. Después de estar en posición, las piezas se ensamblan mediante un golpe, permaneciendo unidas gracias a la deformación plástica generada en el material.

Para la organización de las piezas en el ensamble de botones metálicos de golpe se utilizan tambores vibradores, los cuales consisten de una bobina generando la vibración, y una estructura con vigas en ángulo, donde actúan como resortes para que la vibración del tambor ocurra en distintos ejes. De esta forma las piezas dentro del tambor se movilizan de una forma controlada y circular. Del tambor, o tolva vibradora las piezas pasan a un riel de ensamble, el cual permite el paso solo de una pieza a la vez para luego ser ensambladas con un golpe.

La visión artificial es una disciplina que integra todos los elementos proporcionados a una máquina para que pueda realizar tareas que desempeña el ojo humano. Ésta se encarga de la captura de imágenes, con el fin de extraer las características y propiedades de

objetos, para finalmente hacer una interpretación que satisfaga una necesidad del usuario (González Marcos, y otros, 2006).

El procesamiento de imágenes consiste en algoritmos que mejoran la calidad de las imágenes para poder ser analizadas, debido a los defectos que puedan aparecer en la captura de la imagen (Enrique Sucar & Giovani Gomez, 2011). Una segmentación es una tarea utilizada en el análisis de imágenes que tiene como objetivo dividir la imagen en partes para diferenciar los píxeles de un objeto u otro (Guindos Rojas, Piedra Fernández, & Peralta López, 2001). La binarización de una imagen es una tarea que convierte cada pixel de la imagen en blanco o negro, dependiendo de un umbral determinado previamente (Guindos Rojas, Piedra Fernández, & Peralta López, 2001). Filtrar una imagen consiste en aplicar un proceso para obtener una nueva imagen con diferentes características, generalmente evitando el ruido e imperfecciones de la misma. (Enrique Sucar & Giovani Gomez, 2011).

Para la identificación de objetos mediante visión artificial son generalmente utilizados sistemas de entrenamiento. Para ello es necesario tomar aproximadamente 50 fotos del objeto a identificar y hacer un análisis y procesamiento de las imágenes para extraer y almacenar características claves de los objetos a identificar como el área, perímetro, color, excentricidad, entre otros. Cuando se tienen estas características del objeto en el sistema es posible comenzar a generar una identificación de los objetos de forma automática mediante algoritmos (Cantero & Martinez, 2013).

Machine Learning, o el aprendizaje automático es una rama en evolución de los algoritmos computacionales diseñados para emular la inteligencia humana al aprender del entorno circundante. Una gran cantidad de datos son introducidos en los algoritmos para que a partir de ellos se puedan desarrollar funciones con la capacidad de determinar diferencias entre entradas de datos (El Naqa & Murphy, 2015).

2 METODOLOGÍA

2.1 Diseñar un dispositivo de identificación y remoción que cuente con un sistema de detección por visión artificial y un separador de piezas en la línea de ensamble

Lo primero que se debe hacer en una aplicación de un proyecto de visión artificial es determinar el entorno en el que se va a trabajar. Esto incluye los materiales con los que se va a realizar el proyecto, al igual que el lugar en donde se planea implementar el dispositivo. Al seguir este orden de ideas, antes de saber con qué cámara se va a trabajar es necesario conocer el entorno en el que va a estar sometida la cámara para que la elección de equipo sea la más óptima y eficiente en el proceso.

El dispositivo de visión artificial va a actuar en un tambor vibrador en el ensamble mecánico de los botones metálicos de golpe. Dicho tambor se encuentra en la fábrica de la empresa Estrada Velásquez, en un ambiente industrial donde la interacción con operarios y personal, ruido, máquinas y materiales de ensamble es constante. La iluminación en el área de trabajo es variable ya que la luz en la bodega es dependiente del clima y del ambiente, y las sombras que se pueden generar por las personas que transiten en cercanía del tambor. Debido a esto una buena práctica en cuanto a la implementación de sistemas de visión artificial es el aislamiento de la zona de trabajo lo más que se pueda, para evitar posibles errores en la operación y mantener el estado del entorno de trabajo controlado y constante.

Las principales variables que determinan la elección de la cámara son el costo, la velocidad de visualización de la cámara, la resolución de la cámara, la distancia de enfoque, la luminiscencia del entorno, la vibración del tambor y su ubicación. Más adelante en el trabajo se discute la importancia de cada una de las variables a la hora de tomar una decisión a partir de ellas.

El sistema de remoción de la pieza de la línea de ensamble debe ser diseñado teniendo en cuenta el entorno en el que va a trabajar. Se planea realizar diseños experimentales del sistema de remoción, comenzando desde los diseños más básicos y sencillos hasta los más complejos en caso de ser necesario. Esto se hace para evitar el sobredimensionamiento de las soluciones y evitar gastar recursos que pueden no ser completamente necesarios en la realización de las piezas.

En el entorno de trabajo se cuenta con líneas de aire comprimido, lo que da la posibilidad de utilizar o considerar éste como medio de separación. Adicionalmente el separador de piezas debe ser lo menos invasivo posible de forma que no afecte el funcionamiento actual o futuro de la máquina de ensamble, es decir, que en caso que se desee desmontar el sistema de detección de piezas, sea posible hacerlo fácilmente. Teniendo en cuenta lo anterior, se descarta completamente la posibilidad de soldar las piezas del

separador a la máquina, ya que es un proceso que afecta de forma definitiva los objetos involucrados. Toda actividad de soldadura que se realice no debe involucrar directamente al tambor vibrador o al riel de ensamble.

El acople electromecánico del sistema, donde se convierte la señal digital generada por la unidad de procesamiento en una acción física que debe remover la pieza del área de trabajo y la línea de ensamble, también debe evitar sobredimensionarse, ya que las herramientas e instrumentos que se pueden utilizar en esta etapa del proyecto tienden a resultar costosos. En caso de trabajar con aire comprimido, y considerando la aplicación del trabajo y el hecho de que trabaja con pequeñas piezas metálicas, una válvula electro neumática de solenoide pequeña 2/2 de 24V y de simple acción puede ser suficiente. Para el acople digital, un Arduino es capaz igualmente de presentar los resultados necesarios a bajo costo.

2.2 Desarrollar un algoritmo para el sistema de detección por visión artificial

Los algoritmos utilizados para la detección de las piezas se pueden separar en 4 tareas principales. La primera es la captura de las imágenes para comenzar a trabajar con ellas. La captura inicial de imágenes se hace para conocer el material con el que se va a trabajar. Es importante que en la captura inicial de imágenes se tengan muestras de lo que se quiere y de lo que no se quiere, para poder comenzar a diferenciar claramente entre ambas. El número de imágenes para el sistema de entrenamiento varía con la aplicación, pero de forma general se puede ver que sistemas de entrenamiento tardan entre 20 y 100 intentos para lograr un error despreciable, como el caso de Cantero y Martinez en el que la diferenciación entre útiles escolares tardó 26 intentos. Siguiente a la captura de imágenes se realiza el procesamiento de las imágenes, donde se realizan acciones de pre-procesamiento y segmentación, para limpiar las imágenes de ruidos innecesarios, filtrarlas para un entendimiento más acertado por el computador, y donde se separa e identifica el objeto que se quiere analizar al fondo y demás factores presentes en la toma (Cantero & Martinez, 2013).

Después de un procesado apropiado, se debe realizar un algoritmo de extracción de características, donde de definen las clases que se quieren identificar y cómo diferenciarlas de las que no. Las características extraídas son las que se utilizarán posteriormente para el entrenamiento y serán las razones por la cual el computador pueda tomar decisiones acertadas o no. Finalmente, sigue el entrenamiento del sistema, utilizando las características extraídas para entrenarse en reconocer lo que va en distintas categorías.

El algoritmo de clasificación de las piezas se realiza en Python. En este se realiza un modelo capaz de diferenciar las piezas y sus reveses. Para esto se procede a construir una red neuronal con 50 imágenes por cada tipo de pieza. Las imágenes adquiridas a través de la cámara pasan por una etapa de preprocesamiento, en la cual se adecuan por medio de filtros al eliminar el ruido que se produce en el momento de la captura. Posteriormente la imagen mejorada se binariza, este proceso consiste en definir cada

pixel de la imagen como blanco o negro, según un rango determinado previamente. Después se extraen las características y de esta forma se entrena la red neuronal, con las mismas utilizadas en el pre procesamiento.

Para realizar las pruebas en vivo y verificar el funcionamiento total del sistema, se programa un archivo que tome el video de la cámara, sea capaz de determinar cuándo es el momento adecuado para realizar el procesamiento de la imagen y utilizando los archivos generados, tomar una decisión a partir de lo que se percibe en el instante. El método de detección básico se basa en la identificación de contornos. De estos es posible extraer áreas y posiciones o coordenadas en la imagen, los cuales brindan herramientas adicionales para establecer reglas de detección en vivo. Como regla general, es lógico y apropiado esperar a que la pieza este situada en el centro de la imagen y que no exista movimiento antes de que el sistema comience a procesar.

2.3 Implementar el dispositivo funcional en un tambor vibrador

La implementación del sistema de detección y remoción en el área de trabajo consta en reunir todas las diferentes piezas que componen el dispositivo, y ensamblarlas acorde a su diseño. Para la realización de esta etapa del trabajo es necesario que todas las partes del trabajo estén completadas: física y virtualmente. Esto incluye la base para el sistema de adquisición de imágenes, el separador físico de las piezas en la línea de ensamble, el software o algoritmos necesarios para el funcionamiento del sistema, y las conexiones relevantes para el funcionamiento de cada pieza.

Para el ensamble de las piezas es necesario que la máquina deje de operar en su actividad diaria, y reservar unas pocas horas para realizar la implementación del sistema. Gracias al diseño de las partes físicas del dispositivo, el ensamble de cada pieza se debe poder realizar fácilmente y de una forma no invasiva para la maquina ensambladora de botones. La herramienta principal para el ensamble de las piezas son los tornillos, ya que de estos dependen los diseños de las partes del sistema previamente discutidos. El uso de tornillos facilita el montaje y desmontaje de cada pieza.

Una vez el área de trabajo esté disponible para la implementación, se comienza con el ensamble del sistema de separación de piezas y luego con el sistema de adquisición de imágenes. Finalmente se pueden realizar el resto de conexiones eléctricas, electrónicas y neumáticas para darle la funcionalidad necesaria al sistema. La conexión de la unidad de procesamiento, o el computador utilizado para la ejecución de los algoritmos da cierre a la etapa de implementación, y puede ser considerada una etapa adicional de prueba para el sistema diseñado e implementado. Es importante asegurarse que cada conexión y unión esté realizada exitosamente. Casos particulares para revisar pueden ser la conexión de potencia de los actuadores, las conexiones electrónicas, la presión y conexión del sistema neumático y la firmeza de las piezas mecánicas como tornillos y tuercas.

2.4 Verificar el funcionamiento del dispositivo en la empresa C. I. Estrada Velásquez

La fase final del trabajo consta en comprobar lo realizado a lo largo del trabajo. La fase de verificación final puede ser empezada solo cuando todas las partes anteriores del proyecto se han finalizado y el área de trabajo está instrumentada con el sistema de detección y separación de piezas. En el proceso de diseño y programación las distintas partes del sistema pueden ser verificadas individualmente, pero el resultado estas pruebas no son criterio aun de que el trabajo ha sido exitoso.

La verificación del proyecto se realiza con las tres referencias escogidas en un principio y con las que se hizo cada etapa del trabajo. Para la prueba se tienen aproximadamente 100 unidades de cada referencia de pieza. A la hora de realizar la validación cada referencia de piezas se pone por separado en el tambor vibrador para que el sistema de detección funcione. En este proceso no es necesario que se ensamble la pieza como tal, ya que esto es una etapa de la fabricación de botones en donde el trabajo no tiene influencia, por lo que se puede evitar el gasto de material para realizar la prueba.

Una de las principales características evaluadas para el sistema de detección y remoción es su habilidad para detectar e identificar la pieza en particular que tenga bajo el lente, y la capacidad del software para tomar una decisión entre dejar la pieza en la línea de ensamble y removerla. Esta decisión se puede evidenciar en la ejecución del algoritmo mostrado por la unidad de procesamiento, o con el funcionamiento como tal del sistema de remoción. Si la pieza esta al derecho, no se debe remover la pieza; si la pieza esta al revés, se debe esperar una remoción y una actuación en el sistema de remoción.

La segunda característica a evaluar es la remoción efectiva de la pieza en caso de ser necesario, mediante el juicio del algoritmo. Cuando la señal de remover o no remover la pieza es enviada por la unidad de procesamiento, el funcionamiento del actuador debe ser consecuente con la lógica de operación, al igual que todos sus componentes para su operación. Adicionalmente, la remoción efectiva de la pieza involucra la capacidad del sistema de remoción de sacar de la línea de ensamble solo una pieza. Sacar más de una pieza de la línea de ensamble a la vez implica una ineficiencia en el sistema, la cual debe ser evitada.

El criterio de evaluación es el conteo de acciones correctas que toma el sistema, ya sea por acción o inacción. De esta forma se puede sacar un porcentaje de asertividad, el cual puede ser comparado con el porcentaje que se pueda obtener en la simulación de los algoritmos. Con el porcentaje de efectividad del sistema se puede discutir acerca del éxito del proyecto implementado.

3 PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

3.1 Desarrollo de dispositivo de identificación y remoción de piezas en la línea de ensamble

Inicialmente, las variables tenidas en cuenta para escoger la cámara con la que se realizó el proyecto fueron las siguientes:

Costo

Cómo el dispositivo a implementar hace parte de un proceso exploratorio, es importante para la empresa que le de uso, que no tenga un precio elevado. Para lograr un bajo costo y un buen desempeño es importante considerar la escala del proyecto para que en la selección de materiales no estén sobredimensionados para la aplicación que se les va a dar.

Velocidad de visualización y resolución

La velocidad de percepción del ojo humano es de 24 cuadros por segundo, siendo éste el mínimo que se puede llegar a considerar en una cámara. Velocidades de 120 cuadros o más pueden ser considerados demasiado altos para la aplicación actual, ya que las piezas no se moverán a altas velocidades. Sin tener gran exigencia en los requerimientos y especificaciones de la cámara, la velocidad mínima de visualización debe ser de 30 cuadros por segundo, y una resolución de 0.3MP, suficiente para que la totalidad de las piezas sean visualizadas en la imagen.

Distancia de enfoque

Las piezas que se capturaron en video y para su análisis son pequeñas, con dimensiones variando entre 10 y 20 milímetros aproximadamente, por lo que la cámara debe contar con un buen enfoque a corta distancia o con capacidad de tomar fotografías de tipo macro. En éste tipo de fotos, la cámara puede enfocar objetos a cortas distancias del lente sin perder el nivel de detalle de la pieza como tal. Por las dimensiones de las piezas de la zona de trabajo, la distancia de enfoque no debe ser mayor a 10 centímetros, ya que a distancias superiores a estas, las piezas analizadas se vuelven muy pequeñas para que la cámara pueda captar su detalle, y el ángulo de visión resulta ser demasiado amplio, incrementando así las posibilidades de equivocación del sistema debido al fondo en la toma.

Vibración y ubicación

Por el tambor vibrador las piezas se mueven internamente hasta llegar a su borde, donde las piezas continúan su camino por canales hasta que son alimentadas al mecanismo de ensamble automático. Por la vibración del tambor puede resultar problemática la captura

de las piezas en su trayecto, y es por esto que la cámara fue ubicada en un sitio por fuera del tambor, donde la vibración tuvo un efecto despreciable en la captura de imágenes de las piezas. Al analizar el espacio de trabajo se descubrieron distintos posibles lugares donde puede ir ubicada la cámara. Se resumen las posibilidades a 2, presentadas a continuación.

En el tambor o tolva

El espacio de trabajo se puede ver en la Ilustración 1.

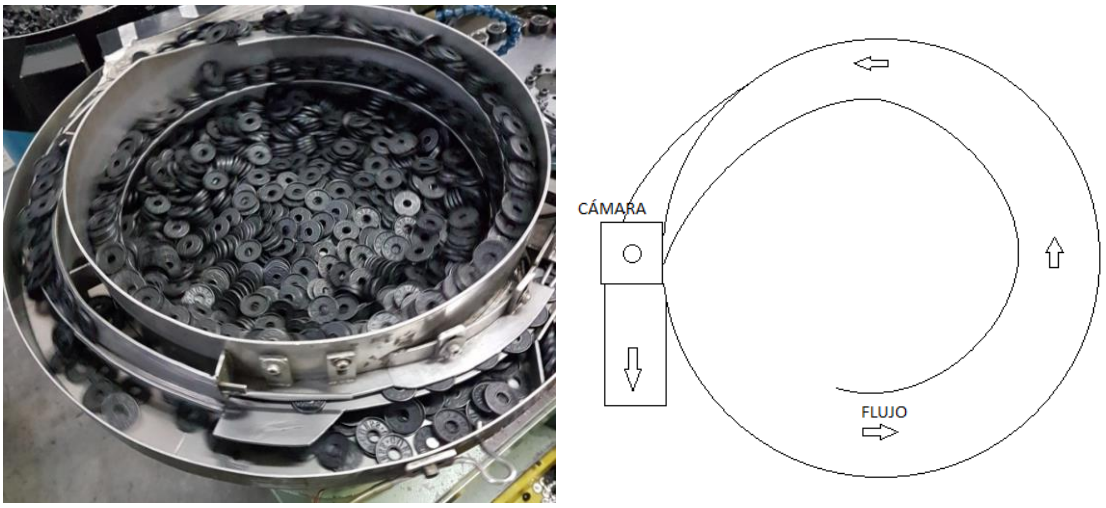

Ilustración 1: Zona de tambor vibrador

Cómo se puede ver, las piezas en esta etapa aún están organizándose y separándose, por lo que es probable tener piezas amontonadas, o unas encima de otras. Adicionalmente, todas las piezas vibran bruscamente. La ubicación de la cámara en este sitio implicaría una serie de problemas debido a esto, ya que la adquisición de la cámara debería ser de alta velocidad. Finalmente, la construcción de la base para la cámara, y su acople son de mayor complejidad en caso de escoger esta ubicación, ya que no se tiene tanto espacio para apoyar una estructura, y no se puede interferir en la operación de la máquina.

En el riel de salida

El espacio de trabajo se puede ver en la Ilustración 2.

Ilustración 2: Zona de riel de ensamble

En la imagen se puede ver donde comienza el riel transportador de piezas para el ensamble. Éste no está físicamente en contacto con el tambor vibrador, y en este punto las piezas logran estar ordenadas una tras otra, facilitando la captura de imágenes. Además hay espacio suficiente para acoplar una estructura sin interferir en la operación de la máquina.

Los canales, o rieles mencionados anteriormente, están físicamente desacoplados del tambor, haciéndolos un lugar óptimo para ubicar la cámara ya que no comparten directamente la vibración del sistema. El comienzo del riel es la posición adecuada donde fue ubicada la cámara. Éste mide aproximadamente 50mm de ancho.

3.1.1 Selección de cámara

Con todo lo anterior en mente, se procedió a seleccionar una cámara útil y con la capacidad de tener un buen funcionamiento para el proceso que va a realizar. Además de ser pequeña, es ideal que cuente con un enfoque manual, para así asegurar que el detalle de las piezas va a poder ser percibido a la corta distancia a la que se debe ubicar la cámara. El trabajo no necesita un equipo demasiado robusto, ya que no se va a realizar una captura de imágenes a muy altas velocidades o en un entorno de poca visibilidad. Las cámaras candidatas se pueden ver en la Tabla 1.

Cámara	Costo	Velocidad de visualización	Resolución	Distancia de enfoque	Luminiscencia
Supereyes B005	$27.99 (USD)	30 fps	640 x 480	0 – 15 mm	4 LED incluidos
Microsoft LifeCam HD 3000	$39.95 (USD)	30 fps	1280 x 720	30 – 150mm	No incluye
Módulo ELP-USBFHD01M-L21	$45.99 (USD)	120 / 60 / 30 fps	640 x 480 / 1280 x 720 / 1920 x 1080	Variable	No incluye

Tabla 1: Comparación de cámaras

El microscopio digital B005 de Supereyes resultó ser una opción viable por su bajo costo en relación con cámaras industriales y por su opción de enfoque manual, resolución y cantidad de cuadros por segundo apropiados. Cuenta también con 4 LEDs blancos integrados en la punta del dispositivo, con control de su intensidad y conectividad USB sin necesidad de instalar software adicional. La información se ve más en detalle a continuación en la Tabla 2.

Característica	Detalle
Magnificación	1-200x
Sensor	0.3MP
Tipo de enfoque	Foco manual
Dimensiones	Diámetro 11mm x 123mm
Resolución	640 X 480
Velocidad de visualización	30 fps
Interfaz	USB

Tabla 2: Especificaciones de Supereye B005

Con la cámara enfocando las piezas a la distancia deseada de 6 centímetros, se verificó que la luz suministrada por los LEDs integrados fuera suficiente para que la pieza se vea completamente visible, tanto en iluminación como en tamaño de la imagen. Efectivamente, la distancia de la cámara a las piezas fue apropiada ya que hay suficiente espacio para visualizar la totalidad de la pieza sin que se pierda resolución por el pequeño tamaño de la misma.

3.1.2 Diseño y construcción de la base de la cámara

A continuación, se procedió a realizar un diseño experimental de la base para la cámara. Teniendo en cuenta las especificaciones de la cámara y sus dimensiones, y el entorno en el que va a trabajar se decide que el material para realizar el prototipo debe cumplir con ciertas especificaciones.

Ya que la zona en la que se va a trabajar está inclinada, el peso de la estructura entra en consideración, porque de ser muy pesada, es posible que sufra una caída o dificulte su acoplamiento. Lo anterior da pie a la siguiente consideración: resulta práctico que la base fuera de fácil montaje y desmontaje, ya que, en el desarrollo del trabajo, mientras se hace el desarrollo de los algoritmos de procesamiento y demás etapas operativas es posible la necesidad de un cambio de hardware o diseño. Adicionalmente, mientras el dispositivo no opere, debe interferir lo menos posible con la máquina y el operario actual que trabaja con ella en la fábrica. Finalmente, la estructura debe ser estable y rígida para que sostenga su correcta operación en un ambiente laboral industrial.

Con lo anterior, se decidió que el material a utilizar es madera MDF de espesor de 6mm por su buena rigidez y facilidad de utilizar. El diseño de la base constó de un prisma rectangular, o un “tubo” de perfil cuadrado formado con 4 lados de madera, y dos soportes internos donde estará sostenida la cámara, también en madera. Se optó por un diseño de este estilo por la geometría de la cámara, que tiene sus dimensiones similares a las de un lápiz. Para el acople de la base con el riel, o la zona de trabajo se diseñó un sistema de abrazadera utilizando placas de metal y tornillos para apretar. Dos placas atornilladas a los lados de la estructura, una a cada lado, junto con una placa al otro lado del riel sujetaron mediante tornillos la base de la cámara. Los agujeros por donde pasa la cámara deben tener una medida muy precisa para que la cámara pueda montarse y desmontarse a presión, aprovechando la naturaleza elástica de la madera.

Teniendo en cuenta las dimensiones del espacio de trabajo y la cámara se procedió a diseñar las partes para el ensamble de la base en el software Solid Edge. El ensamble se puede ver en la Ilustración 3.

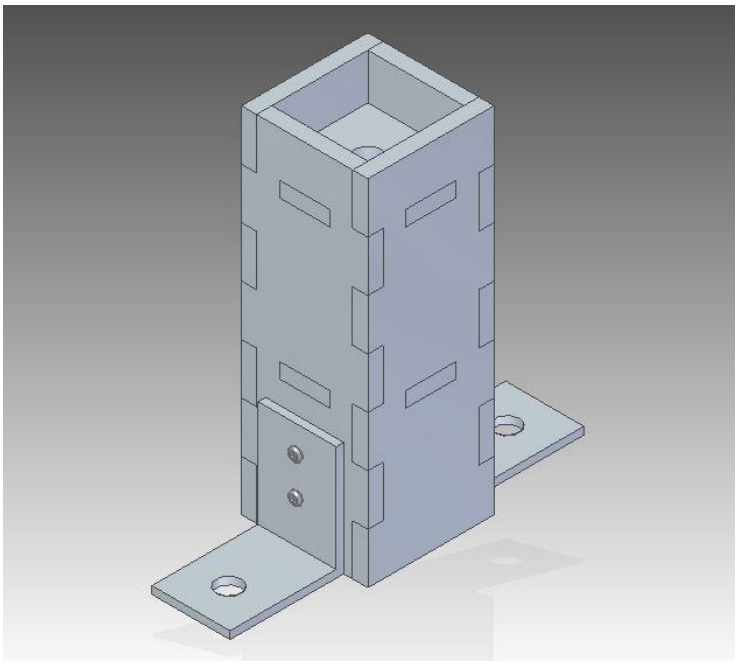

Ilustración 3: Ensamble de la base

Los planos de las piezas fueron luego utilizados para cortar la madera y continuar con el ensamble físico de la base. Los planos en Solid Edge de las piezas para la construcción de la base se pueden encontrar en los archivos adjuntos con el informe. De acuerdo a los mismos, las piezas para el montaje de la base se pueden ver en la Tabla 3.

No.	Pieza
2	LateralMacho.dft
2	LateralHembra.dft
1	PlacaAgujeradaInferior.dft
1	PlacaAgujeradaSuperior.dft
2	PlatinaL.dft

Tabla 3: Piezas para el ensamble de la base

Luego se ensamblaron las partes utilizando un poco de pegamento, se atornillaron las placas laterales y se ubicaron frente a la placa metálica trasera mediante tornillos que se pueden apretar utilizando tuercas. La base resultante fue una capaz de montarse y desmontarse fácilmente, y cumple con las especificaciones para la toma de imágenes,

siendo lo suficientemente rígida gracias al espesor de la madera. Además, el sistema de tornillos para posicionarse a presión en el riel hace que la base se pueda ubicar en sitios con geometrías diversas. El ensamble físico terminado puede ser visto en la Ilustración 4

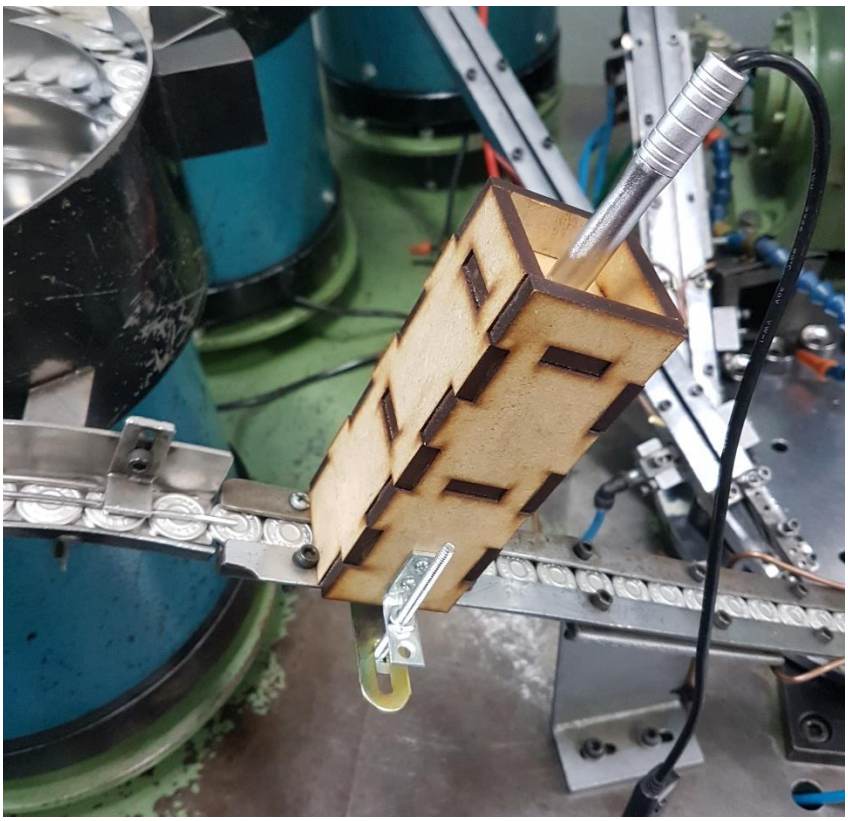

Ilustración 4: Ensamble físico de la base de la cámara con sistema de acople

El sistema de acople resultante tiene la flexibilidad de poder cambiar la posición de la base, y la forma en la que se une con el riel de línea de ensamble. En caso de ser necesario, la platina inferior con la que la base se acopla con el riel puede ser cambiada por más, o distintas platinas en caso de necesitar cambiar la posición de donde aplican presión. Esta flexibilidad fue de enorme importancia para el desarrollo del sistema de remoción, ya que el diseño del mismo no debe necesariamente depender del diseño de la base y su forma de acople.

3.1.3 Diseño y construcción del separador de piezas

A continuación, se presentan las condiciones y los supuestos con lo que se decide comenzar a diseñar de forma experimental el sistema de remoción de piezas en la línea de ensamble.

La regla más importante a la hora de diseñar el separador es que no debe ser muy invasivo en el área de trabajo. En el momento en el que se decida quitar el sistema de la

máquina en la que opera, la máquina debe ser capaz de trabajar exactamente igual a como trabajaba antes de la construcción del sistema de detección y remoción. Además, para el diseño se aprovechó la presencia de una red de aire comprimido en la zona de trabajo de la máquina. En este caso el aire comprimido cumple con el supuesto de un sistema poco invasivo en la separación de las piezas en la línea de ensamble. Estas dos condiciones fueron las principales, además del entorno y las geometrías que tiene la línea de ensamble, con las que se comenzó el proceso de diseño para el separador.

Inicialmente se consideraron dos métodos para extraer la pieza con el uso de aire comprimido, el esquema de las soluciones hipotéticas se puede apreciar en la Ilustración 5. La primera es directamente utilizar la corriente de aire generada para empujar la pieza fuera del riel de ensamble, aprovechando el poco peso de las piezas metálicas, y su geometría con alta resistencia al aire. La segunda solución considerada es utilizar el aire comprimido para accionar un pequeño pistón, y remover la pieza de la línea de ensamble mediante este. Para ambas consideraciones iniciales la pieza resulta ser removida de forma lateral en la zona de trabajo. La remoción en otros sentidos es descartada por practicidad considerando la zona de trabajo y la construcción de la base para el sistema de detección.

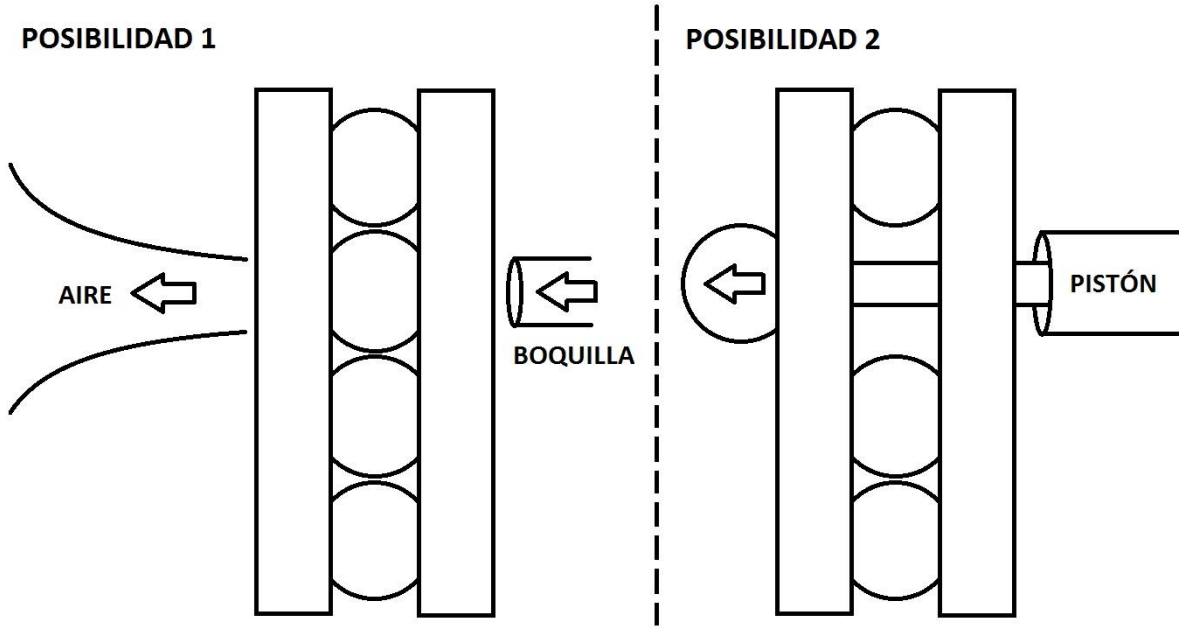

Ilustración 5: Esquema de las opciones para separar piezas

Cómo la opción 1 es más sencilla en su implementación, se procedió con el diseño experimental de un separador siguiendo la forma de operar mostrada en la posibilidad 1 de la Ilustración 5. El par de platinas en el riel son necesarias para que las piezas del ensamble no salgan de su curso, por lo que es una parte de la máquina que no se alteró en la construcción del separador. La parte lateral del riel, sin embargo, se puede modificar para darle entrada al aire que empuja la pieza, y para darle espacio a la pieza que sale del lado contrario. En este caso, el camino por donde sale la pieza no puede dejar que

más de una pieza evacue por el camino. De acuerdo a lo anterior, algún tipo de compuerta debe ayudar a impedir el paso excesivo de piezas en un instante. Las tapas para los botones metálicos son livianas, y como se puede ver en la Ilustración 2, el riel de ensamble cuenta con una importante inclinación. Aprovechando estas dos propiedades, la compuerta para el control de paso de piezas puede ser un segmento de metal con pivote, que abre cuando el aire lo empuja hacia arriba, y retorna a su posición natural por gravedad cuando el aire comprimido deja de ser presente en el proceso.

La materialización de las ideas recién planteadas se puede ver en la Ilustración 6. En este diseño se realiza un agujero a un lado del riel para que entre el aire, y una ranura un poco mayor en longitud que el diámetro de una tapa de botón para que salga por esta al ser empujada por el aire comprimido. La ranura cuenta con una compuerta metálica con pivote para controlar la salida de piezas de la línea de ensamble. Adicionalmente, el agujero por donde entra el aire fue diseñado en una posición clave para el funcionamiento del mecanismo. Fue ubicado aproximadamente en el punto medio entre el centro de la pieza que se debe remover y su borde superior. De esta forma el aire puede empujar lateralmente la pieza, y la corriente de aire está lo suficientemente cerca de la siguiente pieza en la línea como para que no pueda bajar hasta que se detenga la expulsión de aire. Mientras se remueve la pieza, se genera un colchón de aire en el que flota temporalmente la pieza superior, evitando su escape no deseado.

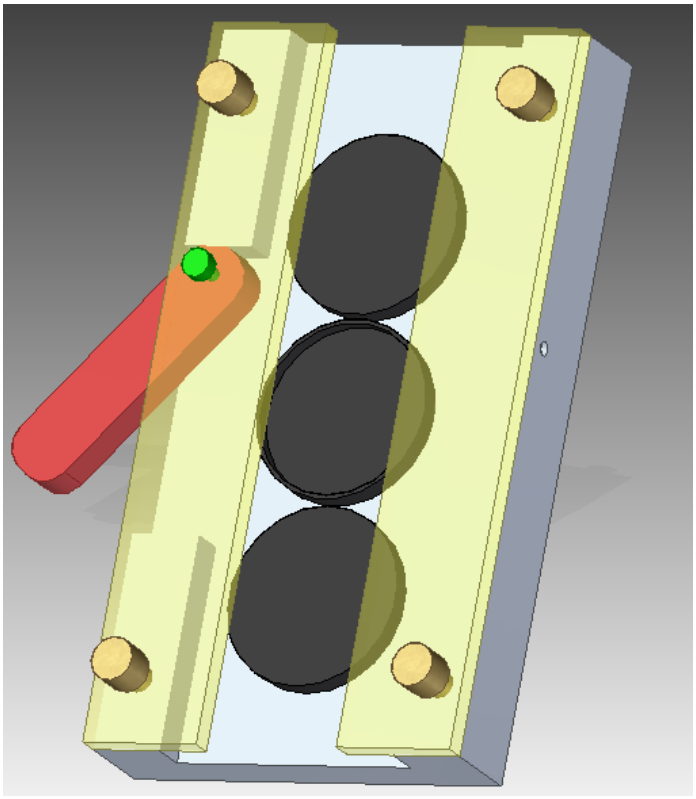

Ilustración 6: Primer diseño conceptual del separador de piezas

Para integrar la ubicación de la fuente de aire comprimido, o la boquilla de la manguera neumática a la solución diseñada, se aprovecharon los tornillos de ensamble del riel, como se pueden ver en la Ilustración 2. Con estos tornillos se puede acoplar una platina debajo del riel que cuente con una boquilla en donde situar la manguera neumática. Adicionalmente, la platina mejoró el diseño anterior de la base para la cámara, ya que de esta platina se puede acoplar mediante tornillos directamente el conjunto entero de detección – separación. La segunda versión del diseño, con la platina que acopla la base de la cámara y la manguera neumática se puede ver en la Ilustración 7.

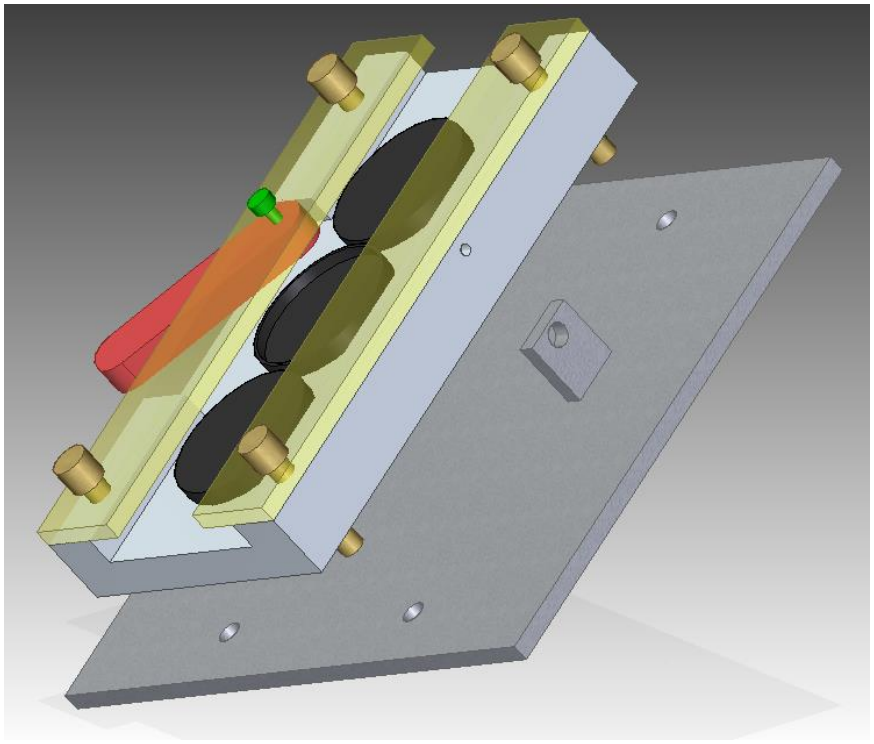

Ilustración 7: Segundo diseño conceptual acoplando el sistema de detección y remoción

Para completar el diseño de remoción de piezas de la línea de ensamble solo hizo falta la conexión entre lo eléctrico y lo mecánico. Como fue discutido anteriormente, la solución viable para utilizar el aire comprimido de forma controlada para actuar en piezas con pequeñas dimensiones, es utilizar una válvula electro neumática de solenoide 2/3 de simple acción y una tarjeta de Arduino Uno como medio de comunicación entre los sistemas.

Para el uso de la válvula mediante el Arduino se utilizó un relay NRP04 de 24 voltios que permite mantener separadas las redes de potencia y de electrónica. La capacidad del relay debe ser apropiada para la potencia que maneja la válvula electro neumática. Como el Arduino Uno solo puede trabajar con señales electrónicas de 0 a 5 voltios y amperajes muy bajos, es necesario utilizar el relay como switch para actual el solenoide de la

electroválvula. Adicionalmente es apropiado utilizar un diodo entre las terminales del solenoide para evitar la corriente en reversa que ocurre con los elementos inductivos al des energizarse. A partir de esto, se diseña un esquema simple de la conexión electrónica que se puede ver en Ilustración 8.

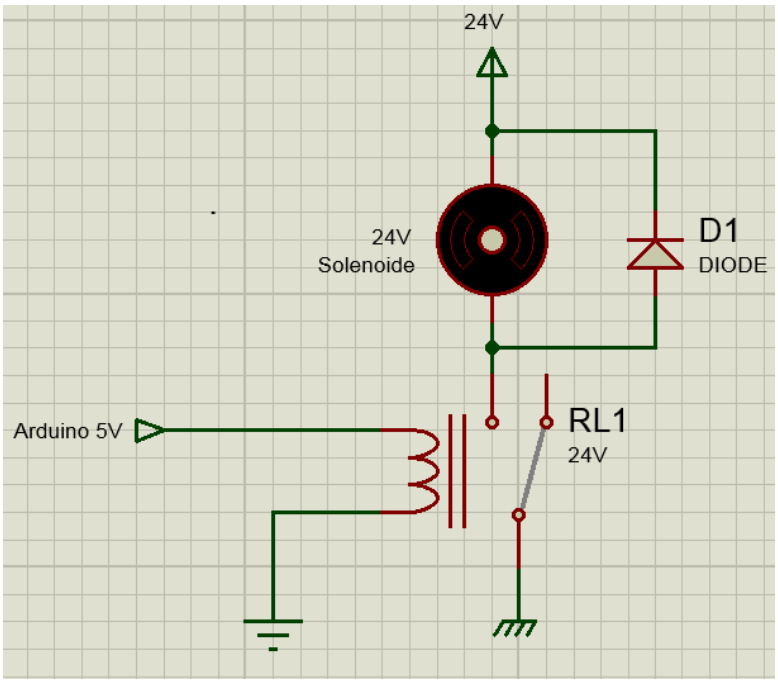

Ilustración 8: Esquema de conexión electrónica y de potencia de la válvula neumática

El programa de Arduino para controlar la válvula es de gran sencillez, ya que su labor en el trabajo es funcionar como un medio de comunicación entre los sistemas, y se puede ver en el Código 1. El micro controlador espera a que se realice una comunicación serial, y dependiendo del valor que se transmita, el microcontrolador envía una señal digital por uno de sus puertos para accionar el relay que funciona como switch para el circuito de potencia. El valor transmitido debe provenir de un script en Python, el cual resulta de la predicción realizada por el sistema de detección.

```
int pin = 13;
void setup() {
  Serial.begin(9600);
  pinMode(pin, OUTPUT);
  delay(300);
}
void loop() {
  if(Serial.available()>0){
    char carac=Serial.read();
    if (carac=='0'){
      Serial.println("Front");
    }
    else if (carac=='1'){
      digitalWrite(pin, HIGH);
      delay(1500);
      digitalWrite(pin, LOW);
      Serial.println("Back");
    }
  }
}
```

Código 1: Script de Arduino para control electromecánico

3.2 Desarrollo del algoritmo de detección

Los algoritmos de detección por visión artificial fueron realizados en el lenguaje de programación Python 2.7, con la ayuda de librerías para la visualización y procesamiento digital de OpenCV. Brosnan y Sun (Brosnan & Sun, 2004) muestran en su recopilación de estudios que en la aplicación de distintas técnicas y modelos de identificación, detección y entrenamiento de sistemas, los factores geométricos son los que resultan tener mejores respuestas, utilizando momentos invariables de las figuras las mayorías de las veces. Los momentos invariables son características geométricas de figuras que no dependen de la

orientación o posición que tengan en las imágenes. Este es el método principal con el que se pensó realizar el algoritmo de detección, utilizando los momentos invariables de Hu. De forma general se puede ver el esquema de diseño para el desarrollo del algoritmo de detección en la Ilustración 9.

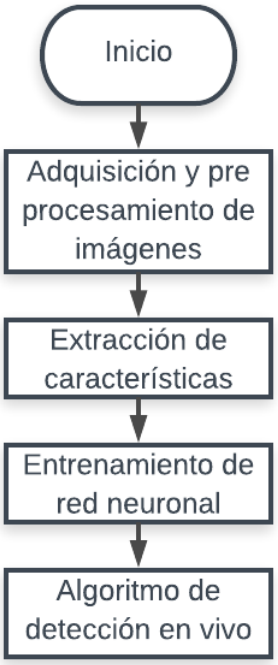

Ilustración 9: Flujo y lógica general de los algoritmos de detección

3.2.1 Captura de imágenes

Para el algoritmo de la captura de las imágenes se realizó una programación en Python para habilitar la cámara, la cual ya estaba conectada y en posición con su base en el área de trabajo. Con la entrada de video habilitada, se creó una variable para tener cada cuadro que recibe la cámara, y un condicional para que tome y guarde una imagen de tipo .JPG cada que se presiona la tecla ‘g’. El código utilizado se puede ver en el Código 2.

```
import numpy as np
import cv2
import time

captura = cv2.VideoCapture(0)

time.sleep(2)
ret, video = captura.read()
i = 0

while(captura.isOpened()):
    ret, frame = captura.read()
    if ret == True:
        cv2.imshow("frame",frame)
    else:
        break
    if cv2.waitKey(1) & 0xFF == ord('g'):
        cv2.imwrite("boton_" + str(i) + ".jpg", frame)
        i = i + 1
        print "boton_ %d" %i
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
#Limpiar memoria y cerrar ventanas
captura.release()
cv2.destroyAllWindows()
```

Código 2: Código de la captura de imágenes

De ésta forma, se tomaron 100 imágenes de distintas piezas en la línea de ensamble, para dar paso al siguiente paso de pre procesamiento de imágenes. En las fotos tomadas estaban también piezas en posiciones no deseadas, o al revés, para que en el procesamiento de la imagen se puedan controlar claramente los procesos que se le van a aplicar a las imágenes para identificar correctamente los objetos. De las 100 fotografías tomadas de cada referencia, 50 son de piezas en la posición correcta y 50 son de piezas al revés. Conociendo como se ven las piezas en sus orientaciones deseadas y no deseadas le permite al algoritmo diferenciarlas.

3.2.2 Pre-procesado de imágenes

Para lograr diferenciar el objeto que se va a analizar del fondo y el ruido en la imagen es necesario realizar un pre procesado. Cada caso de aplicación es distinto, por lo que la manera en la que se puede pre procesar es completamente distinta en cada caso y por ende es necesario analizar la imagen antes de saber cómo actuar. Las imágenes a procesar se pueden ver en la Ilustración 10.

Ilustración 10: Imágenes de una referencia de botón de frente y al revés, antes de ser procesado

Como se tiene tanto brillo y reflejo en la imagen, es difícil separar la pieza del resto de la imagen por completo. Para comenzar, se trabajó con la primera referencia, intentando suavizar la imagen lo más posible sin eliminar características importantes de la pieza metálica. A la imagen en sus colores naturales, en formato RGB, se le aplicaron inicialmente distintos filtros para ver el comportamiento de los mismos sobre la pieza. Los filtros preliminares aplicados fueron Gaussiano, Blurr, Median blurr y Bilateral. Todos estos filtros están incluidos en las librerías de manejo y procesamiento de imágenes que contiene OpenCV. Al analizar el comportamiento de cada filtro se tomó la decisión de utilizar el Bilateral, ya que es la técnica de filtrado que más elimina el ruido y suaviza la imagen sin generar efectos de desenfoque donde se pierde información importante a la hora de realizar una identificación. El efecto de los distintos filtros se puede ver en la Ilustración 11. Como se puede ver, el filtro Guassiano y Blurr distorsionan demasiado la imagen y le quita información importante, mientras que el Median Blurr la altera muy levemente. El filtro bilateral suaviza la imagen sin distorsionarla demasiado.

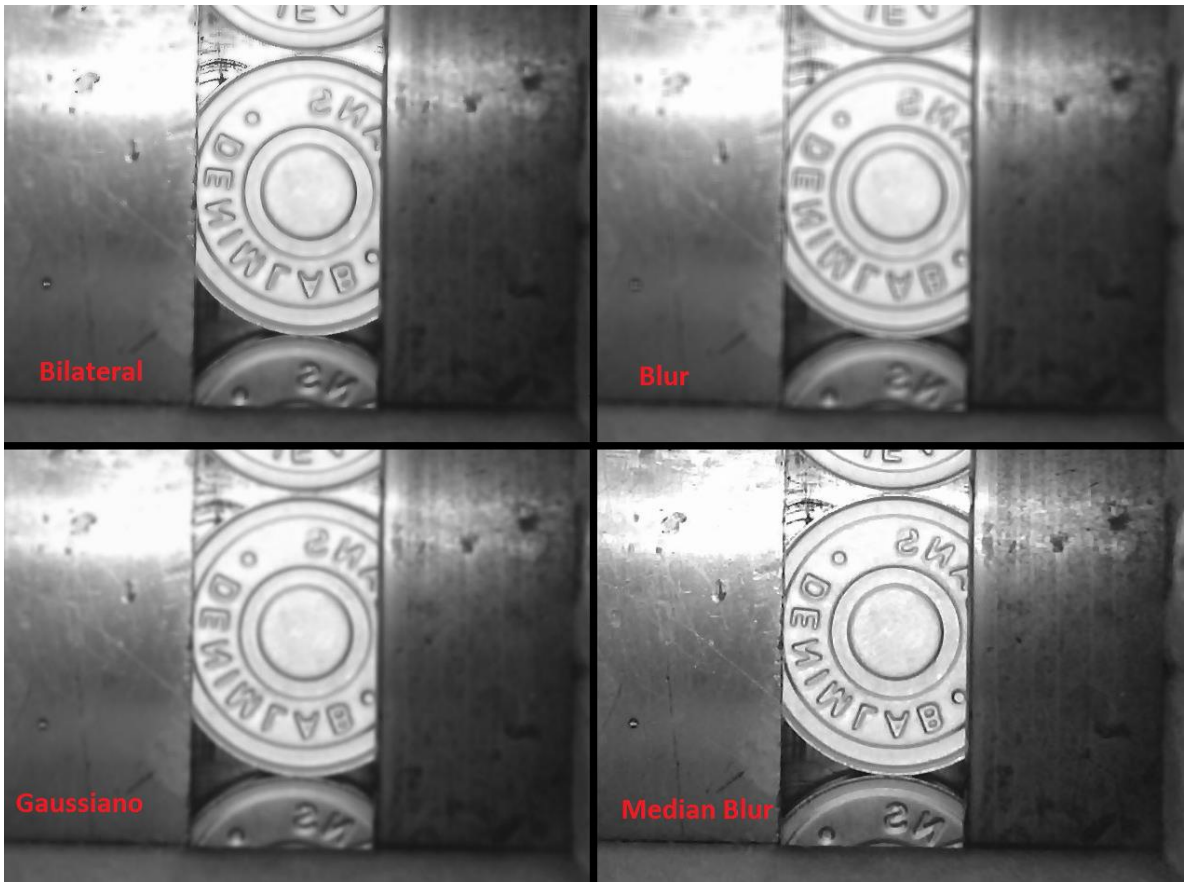

Ilustración 11: Ejemplo de los distintos filtros aplicado a una imagen

Con esta nueva imagen suavizada se continuó pasando la imagen a escala de grises para su posterior binarización. Es necesario realizar el pre procesamiento de la imagen de esta manera ya que la binarización solo se puede aplicar a imágenes que ya estén en escala de grises, cambiando los pixeles a blanco o negro, dependiendo de un umbral definido. Para encontrar el valor del umbral necesario para la imagen se realizó un histograma, donde se puede ver la cantidad de pixeles presentes en la figura de cada tonalidad de gris. Con esto se puede obtener un rango de valores de tonalidad donde se encuentra la mayoría de los pixeles de la imagen. El histograma se puede ver en Ilustración 12, y se puede ver el conjunto de pixeles sobresalientes del común alrededor de 170. Los picos que se encuentran en esta zona son más significativos que los picos encontrados en los extremos de la escala de grises, ya que estos representan los negros y blancos de la imagen, mientras que los picos encontrados en el medio describen las tonalidades grises. El script auxiliar para la realización del histograma se puede ver en el Código 3.

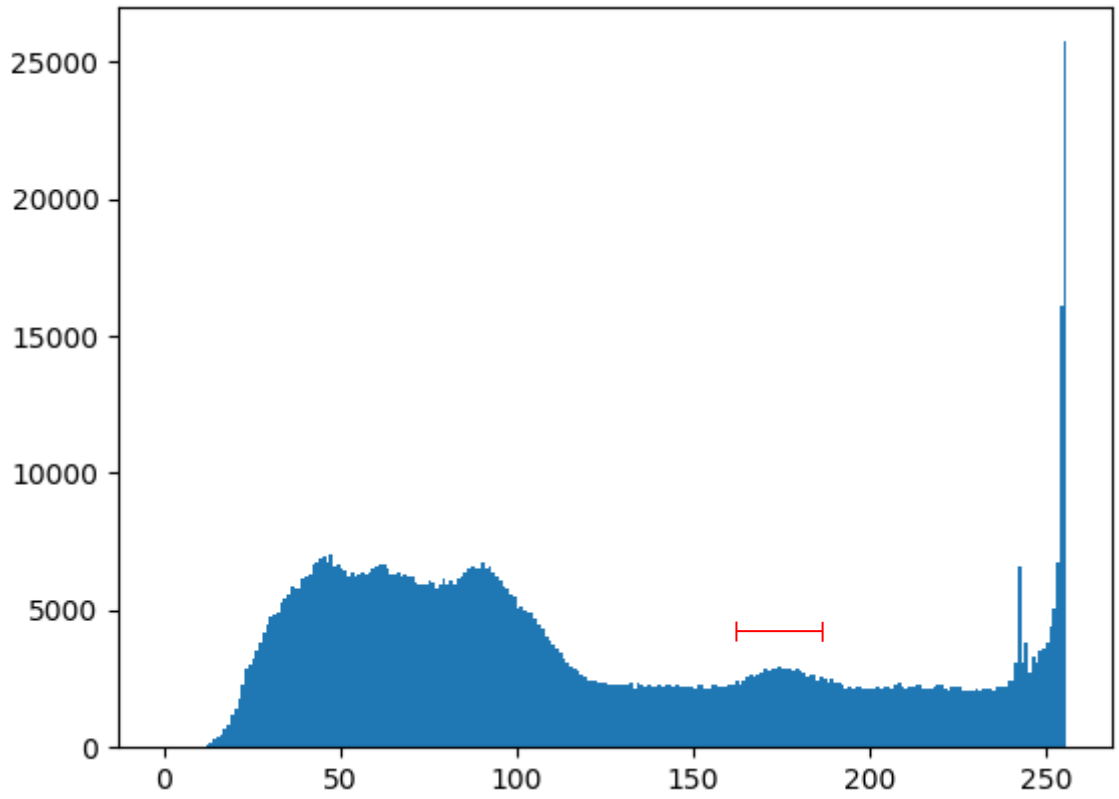

Ilustración 12: Histograma de la imagen

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread("Ref1.jpg")
gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBilateral = cv2.bilateralFilter(gray,9,75,75)

hist = cv2.calcHist([imgBilateral],[0],None,[256],[0,256])
plt.hist(img.ravel(),256,[0,256]);
plt.show()

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Código 3: Script auxiliar para realizar el histograma

Al conocer un posible valor del umbral para realizar la binarización se continuó con el proceso de pre procesado. El valor del umbral, o *threshold*, encontrado no necesariamente garantiza un proceso adecuado de binarización, ya que depende de la totalidad de la imagen. Sin embargo, es un valor por donde se puede comenzar a probar experimentalmente y ver el comportamiento de la imagen. Después de probar con distintos valores se decide optar por 170, ya que es suficiente para que se distinga el botón en la imagen. Ésta se puede ver en la Ilustración 13. El objetivo de la binarización es un procesamiento más rápido y sencillo, para posteriormente lograr encontrar contornos de los cuales se puede diferenciar la pieza de interés del resto del cuadro. El código utilizado para el pre procesamiento de la imagen se puede ver en el Código 4.

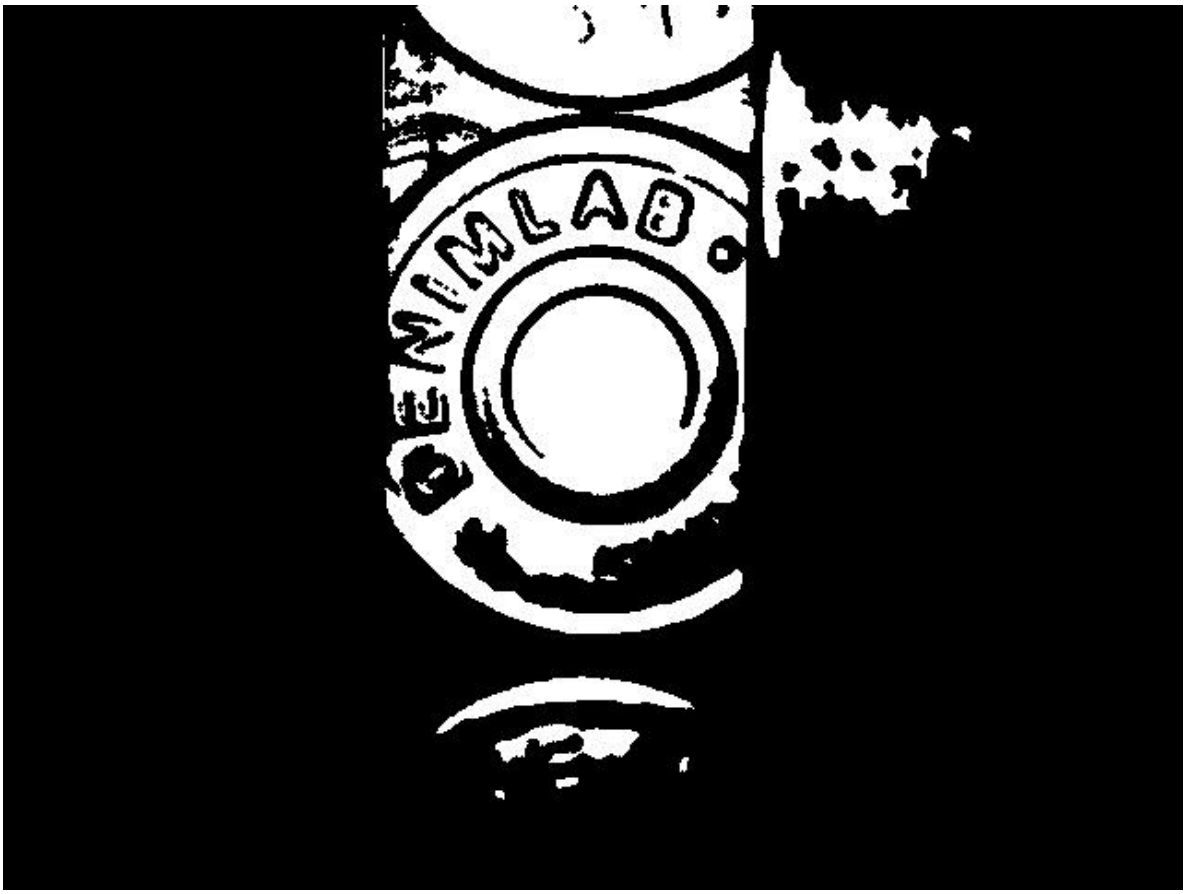

Ilustración 13: Muestra de la imagen pre-procesada

```
import numpy as np
import cv2
from time import time

t0 = time()
path = "Ref1.jpg"
img = cv2.imread(path)
img2 = cv2.imread("Ref1 C.jpg",0)

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))

gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBilateral = cv2.bilateralFilter(gray,9,75,75)
ret, imgBlack = cv2.threshold(gray,170,255,cv2.THRESH_BINARY)

h,w = imgBilateral.shape[:2]
imgBilateral[:h,:205] = 0


ret, imgBilBlack = cv2.threshold(imgBilateral,170,255,cv2.THRESH_BINARY)

erosion = cv2.erode(imgBilBlack, kernel=kernel, iterations=1)

cv2.imshow("Imagen filtro",imgBilateral)
cv2.imshow("Imagen binarizada", imgBlack)
cv2.imshow("Imagen binarizada con filtro",imgBilBlack)
cv2.imshow("Imagen final",erosion)

print time() - t0
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Código 4: Script de pre-procesamiento de imagen

En el desarrollo del Código 4 se utilizaron las imágenes de una referencia de frente y de espaldas para comprobar en cada paso el comportamiento del software. El desarrollo de los scripts realizados para este tipo de aplicaciones tiene una modalidad iterativa y experimental, ya que cada imagen es distinta y no se tiene una forma concreta o estándar de realizar las cosas. Al principio se inicializó una variable para el kernel, el cual es el elemento estructurado que cambia la forma en la que se realizan los procesos de erosión o dilatación. En la inicialización se especifica la forma y el tamaño del kernel que se utilizará para modificar la imagen, en el caso se optó por una circular de 3 pixeles de radio por la naturaleza concéntrica de las piezas analizadas.

A continuación, se realizó el cambio de escala de colores a gris para realizar posteriormente el filtrado bilateral y la binarización de la imagen. Finalmente, la imagen binarizada tiene un proceso de erosión, donde se espera cubrir la mayoría de las manchas presentes en la imagen. El objetivo principal del script es experimental e iterativamente hallar los distintos valores posibles de los filtros, erosiones, dilataciones y kernels para obtener una imagen óptima. El tiempo que tarda en operar la máquina se muestra al final para tener una idea de la velocidad de procesamiento del sistema.

3.2.3 Procesado de imágenes

Teniendo una imagen binarizada, dónde se es capaz de diferencial visualmente los elementos que la componen, se empezó con el siguiente paso de encontrar los contornos. Las librerías de OpenCV tienen funciones especializadas para realizar precisamente eso, encontrar los diferentes contornos en la imagen. Como en la nueva imagen en blanco y negro se tenían tantas manchas y ruido perteneciente al fondo en el área de trabajo, es normal que la función retorne un gran número de contornos distintos hallados por la máquina. Es por esto que fue necesario realizar un código auxiliar, que se puede ver en el Código 5, para identificar el número de contornos y la forma en la que la función los encuentra y los clasifica. Como el botón hace parte de la mayoría de la figura, el contorno generado a partir de este debería ser el más grande. Se utilizó esta teoría para encontrar el contorno y el área en la imagen para analizar, ya que es en esta área donde se debe encontrar el botón.

```
import numpy as np
import cv2
from time import time
from glob import glob

t0 = time()

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
imgCarpeta = 'B0/boton_*.jpg'
imgNames = glob(imgCarpeta)

for n in imgNames:
    img = cv2.imread(fn)
    gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    imgBilateral = cv2.bilateralFilter(gray,9,75,75)
    ret, imgBlack = cv2.threshold(gray,170,255,cv2.THRESH_BINARY)

    h,w = imgBilateral.shape[:2]
    imgBilateral[:h,:205] = 0

    ret, imgBilBlack = cv2.threshold
                               (imgBilateral,170,255,cv2.THRESH_BINARY)

    erosion = cv2.erode(imgBilBlack, kernel=kernel, iterations=1)

    contours, hierarchy = cv2.findContours
                         (eros.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

    dibujo = np.zeros(img.shape,np.uint8)
    i = 0

    for cnt in contours:
        x,y,w_,h_ = cv2.boundingRect(cnt)
        print w_*h_
        if i>250:
            i=0
        cv2.drawContours(dibujo,[cnt],0,(255-i,255,i),1)
        i += 10

        cv2.imshow("Contornos",dibujo)
        cv2.waitKey(0)
print time() - t0
cv2.destroyAllWindows()
```

Código 5: Script auxiliar para la identificación y verificación de contornos

En el Código 5 se comenzó por inicializar y plantear el pre-procesamiento de las imágenes con los métodos y valores ya encontrados en una etapa anterior experimental. Como se conocen los filtros y las modificaciones que se le deben aplicar a las imágenes para que resulten óptimas para el procesamiento, se continuó con la programación del procesamiento como tal. Las distintas imágenes de las piezas con las que se va a realizar el entrenamiento se encontraban en carpetas llamadas B0 y B1 para las piezas al derecho y al revés, respectivamente. El código recorre estas carpetas leyendo cada imagen, pre-procesándola, identificando todos los contornos presentes y dibujando estos contornos sobre la imagen para su visualización. En cada imagen se mostró también el área de cada contorno que encuentra, para conocer el comportamiento total de la imagen. Un ejemplo de la visualización que se dio se puede ver en la Ilustración 14.

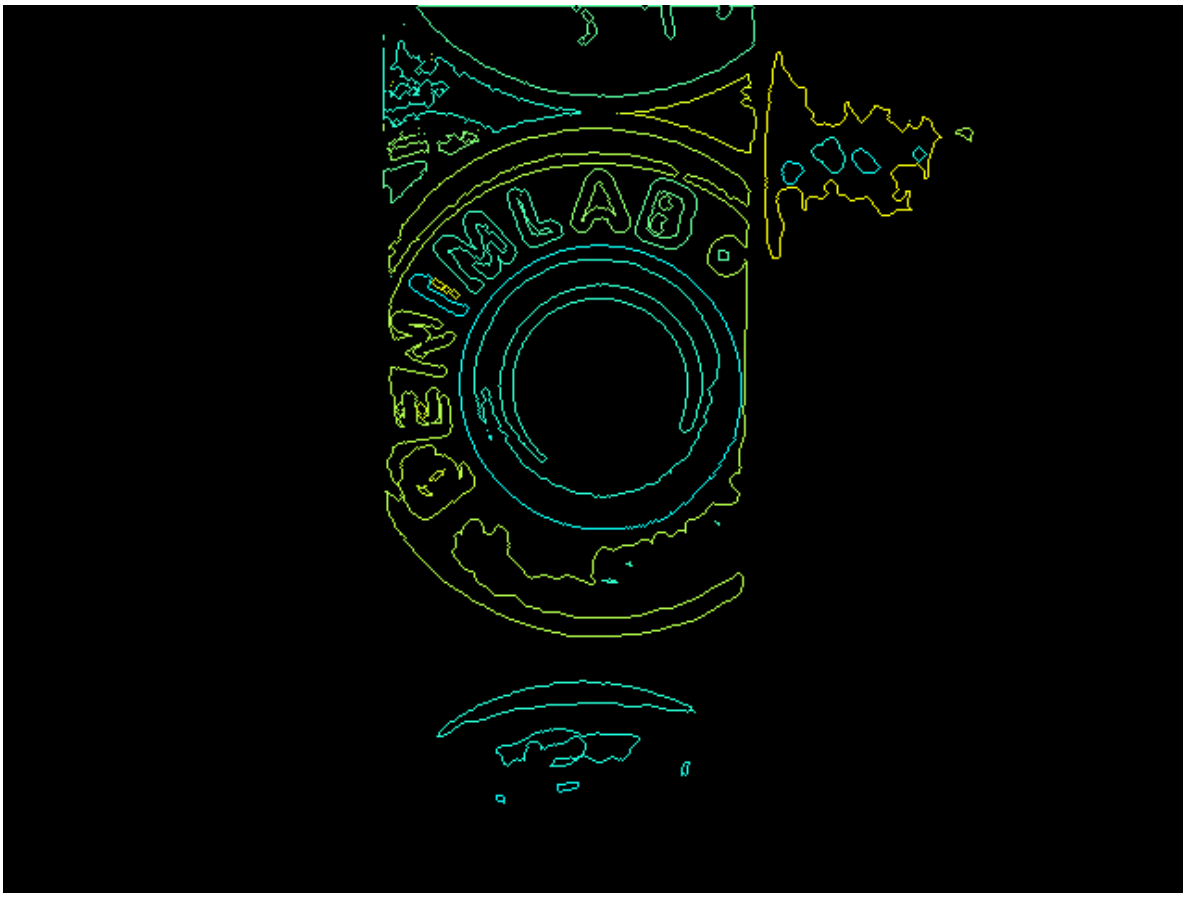

Ilustración 14: Visualización de los distintos contornos hallados en las imágenes

Este código sirve para confirmar la forma en la que el sistema halla los contornos de la imagen, y cuanto miden respectivamente. De esta forma se identificaron varios comportamientos de todas las imágenes, como el hecho de que casi siempre el contorno con mayor área es uno que puede representar la forma del botón y el grabado que lleva

de la marca. Se pudo identificar también, que muchas veces el mayor contorno pertenecía a uno de los reflejos generados por la zona de trabajo. Esto es un obstáculo en la identificación de la pieza, por lo que se optó por quitar el reflejo de la imagen, restando los pixeles de la zona en la que se encuentra el reflejo. Ésta modificación de código se hizo en el Código 4 y se tuvo que hacer en el resto de scripts para mantener homogeneidad en la captura de imágenes.

Dada la experimentación con el código, se pudo identificar que el área del contorno que expresa la forma y marca de la pieza tiene un tamaño cercano a los 50 mil pixeles, y que ningún otro contorno se acerca a un valor como este. Estos valores son mostrados en cada iteración de las imágenes. Ya que el contorno de interés para analizar corresponde a la forma del botón, y el botón siempre es el objeto más grande de la imagen, el contorno al que finalmente se le realizo el análisis es a el que contenga un área mayor. Esta estrategia puede resultar de gran utilidad ya que es flexible a las distintas áreas que puedan llegar los contornos encontrados.

Conocido el contorno de interés se continuó con la extracción de características. En el caso en cuestión, se utilizaron inicialmente los momentos de Hu. Mediante una de las funciones de OpenCV se extrajeron los primeros siete momentos invariables de Hu, los cuales describen la geometría y la excentricidad de los contornos procesados. Estos momentos son factores y pesos promediados que cuantifican la excentricidad y forma de los contornos, con información sobre la escala, rotación y traslación de las geometrías analizadas. La ventaja de momentos es que hace que las geometrías analizadas no dependan de la orientación o escala con las que se capturen, haciendo que el algoritmo sea finalmente más robusto. El cálculo de estos momentos invariables se hace con los centroides encontrados relativo a diferentes ejes.

Los momentos, el área, el perímetro y la relación de escala, que son los criterios para la toma de decisiones, se guardaron en un archivo de Excel para su luego utilización en el entrenamiento de las redes neuronales. En el archivo de Excel, cada columna de la hoja representa una característica distinta del contorno. El código con el que se hizo el procesamiento de las imágenes, y la extracción de las características se puede ver en el Código 6, a continuación:

```
import numpy as np
import cv2
from glob import glob
import xlswriter

row = 0
col = 0

i = 1
c = 0

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))

workbook = xlswriter.Workbook('Datos_Ref1.xlsx')
worksheet = workbook.add_worksheet('Referencia1')

for numFolder in range(0,2):
    carpeta = 'B' + str(numFolder) + '/boton_*.jpg'
    names = glob(carpeta)

    for fn in names:
        print 'procesando %s...' %fn
        img = cv2.imread(fn)
        gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        imgBilateral = cv2.bilateralFilter(gray,9,75,75)
        ret, imgBlack = cv2.threshold(gray,170,255,cv2.THRESH_BINARY)

        h,w = imgBilateral.shape[:2]
        imgBilateral[:h,:205] = 0


        ret, imgBilBlack = cv2.threshold
                              (imgBilateral,170,255,cv2.THRESH_BINARY)

        erosion = cv2.erode(imgBilBlack, kernel=kernel, iterations=1)
        dibujo = np.zeros(eros.shape,np.uint8)

        contours, hierarchy = cv2.findContours
                        (eros.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

        hierarchy = hierarchy[0]
```

```
            # A contornos encontrados, obtener momentos de Hu
            cnt = max(contours, key = cv2.contourArea)
            x,y,w,h = cv2.boundingRect(cnt)
            M = cv2.moments(cnt)
            A = cv2.contourArea(cnt)
            p = cv2.arcLength(cnt, True)
            Comp = A / float(p * p)
            RA = w / float(h)

            Hu = cv2.HuMoments(M)

            print A

            vectorCarac = np.array([A,p,Comp,RA,Hu[0][0],Hu[1][0],
                            Hu[2][0],Hu[3][0],Hu[4][0],Hu[5][0],Hu[6][0]])

            for carac in(vectorCarac):
                if numFolder == 0:
                    worksheet.write(row,0,"Front")
                    worksheet.write(row,i,carac)
                    i = i + 1
                elif numFolder == 1:
                    worksheet.write(row,0,"Back")
                    worksheet.write(row,i,carac)
                    i = i + 1
            i = 1
            row += 1
workbook.close()
```

Código 6: Script para el procesamiento de imágenes y extracción de características

En el Código 6 se utilizó la librería de manejo de archivos de Excel xlswriter, con la cual se creó un archivo y una pestaña para almacenar la información de los contornos encontrados en las imágenes. Mientras se recorren las carpetas donde están contenidas las imágenes, también se realizan los procesos mencionados anteriormente de acondicionamiento y pre procesamiento de la imagen. A continuación, se aplicó el filtrado de contornos por su tamaño, como definido anteriormente, el de mayor área. Se utilizó este procedimiento para solo captar los contornos de la imagen más grandes y representativos.

Para el contorno hallado, se dibuja efectivamente qué contorno se ha identificado y se le extraen sus características para ser escritas en el archivo de Excel. Las características asociadas a la categoría “Front” son correspondientes a los botones en la posición deseada, y la categoría “Back” representa los botones que se encuentran al revés y por

ende deben ser removidos. El vector de características con el que se realizó el entrenamiento del sistema se compone de 11 distintos factores geométricos, los cuales son el área, perímetro, centroides en X y Y, y los siete distintos momentos invariantes de Hu.

3.2.4 Entrenamiento del sistema

Una vez se tuvieron los vectores de características, en el caso del trabajo están en la forma de un archivo de Microsoft Excel, se continuó con el entrenamiento del sistema. Como mencionado anteriormente, las características de selección para el entrenamiento son las geométricas que tenían los contornos hallados. Con la combinación de características se procede a entrenar a una red neuronal utilizando MLPClassifier (Multi-layer Perception). Las funciones utilizadas se encuentran en las librerías adicionales de Sci-kit learn de Python. En el código para el entrenamiento, se probaron dos formas de pre-procesar los datos, o las características, mediante las librerías de *Standard* y *Robust Scaler*. El procedimiento se puede ver en el Código 7. Para ambos métodos de entrenamiento se mostraron los resultados, predicciones y matrices de confusión.

```
# -*- coding: cp1252 -*-
from __future__ import print_function, division
import numpy as np
import cv2
import xlrd
from time import time
#Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.externals import joblib
from sklearn.model_selection import cross_val_score
book_excel = xlrd.open_workbook("Datos_Ref1.xlsx")
def load_xlsx(xlsx):

    sh = xlsx.sheet_by_index(0)
    x = np.zeros((sh.nrows,sh.ncols-1))
    y = []
    for i in range(0, sh.nrows):
        for j in range(0, sh.ncols-1):
            x[i,j] = sh.cell_value(rowx=i, colx=j+1)

        if(sh.cell_value(rowx=i, colx=0)=='Front'):
            y.append(0)
        elif(sh.cell_value(rowx=i, colx=0)=='Back'):
            y.append(1)
    y = np.array(y,np.float32)
    return x,y

##### Inicio del programa #####
if __name__ == '__main__':

    t0 = time()
```

```
# Cargar datos desde un archivo .xlsx
# la función retornará el número de muestras obtenidas y su respectiva
# clase
# Realizar entrenamiento con un modelo Standard
X, Y = load_xlsx(book_excel)
standard_scaler = StandardScaler()
X_S = standard_scaler.fit_transform(X)
# Realizar entrenamiento con un modelo Robust
XX, YY = load_xlsx(book_excel)
robust_scaler = RobustScaler()
X_R = robust_scaler.fit_transform(XX)
# Se separan los datos: un % para el entrenamiento del modelo y otro
# para el test
samples_train, samples_test, responses_train, responses_test =
                            train_test_split(X_S, Y, test_size = 0.3)
samples_train1, samples_test1, responses_train1, responses_test1 =
                            train_test_split(X_R, YY, test_size = 0.3)

mlp = MLPClassifier(activation='tanh', hidden_layer_sizes=(15,15),
                                                max_iter=1000, tol=0.0001)
mlp1 = MLPClassifier(activation='tanh', hidden_layer_sizes=(15,15),
                                                max_iter=1000, tol=0.0001)

mlp.fit(samples_train, responses_train)
mlp1.fit(samples_train1, responses_train1)

response_pred = mlp.predict(samples_test)
response_pred1 = mlp1.predict(samples_test1)

joblib.dump(mlp, "MLP_Standard.sav")

mlp_matrix=confusion_matrix(responses_test,response_pred)
print(mlp_matrix)

joblib.dump(mlp, "MLP_Robust.sav")
mlp_matrix2 = confusion_matrix(responses_test1,response_pred1)
print(mlp_matrix2)
```

```
    print ("accuracy_score: ", mlp.score(samples_test, responses_test))
    print ("response pred: ", response_pred)
    print ("\n")
    print ("done in %0.16fs" % (time() - t0))
    print ("\n")
    print ("accuracy_score1: ", mlp1.score(samples_test1, responses_test1))
    print ("response pred1: ", response_pred1)
    print ("\n")
    print ("done in %0.16fs" % (time() - t0))

scores = cross_val_score(mlp,X_S,Y,cv=10,scoring='accuracy')
print (scores)
print (scores.mean())
print("\n")
scores = cross_val_score(mlp1,X_R,Y,cv=10,scoring='accuracy')
print (scores)
print (scores.mean())

joblib.dump(standard_scaler, "Standard.sav")
joblib.dump(robust_scaler, "Robust.sav")
```

Código 7: Entrenamiento del sistema mediante modelos Standard y Robust Scaler

En el Código 7 se comenzó por importar las librerías que se utilizan en el desarrollo del proceso. Las librerías contienen funciones útiles para el manejo de redes neuronales y de los datos utilizados para entrenar las mismas. Entre los paquetes de sklearn están los del subconjunto metrics, los cuales sirven para analizar y validar los modelos desarrollados.

A continuación, se hace una función para cargar el archivo donde se encuentran las características y asignar los valores de las columnas y filas a los dos principales componentes para iniciar el proceso del entrenamiento de la red: la matriz de datos de entrada y el vector de respuestas, o salidas. La matriz *X* y el vector *Y* representan la forma en la que la máquina asocia las características de cada botón con la respectiva clasificación que se le debe dar. Los datos de la matriz *X* deben estar primero normalizados antes de que sean procesados por la red neuronal, por lo que se utilizaron los escaladores *Standard* y *Robust*, ya que ambos pueden llegar a representar un comportamiento distinto en el clasificador. Naturalmente, el modelo de normalización que presente mejores resultados es el utilizado para continuar el proceso. Mientras que el *Standard Scaler* (SS) promedia y normaliza los todo el conjunto de datos, el *Robust Scaler* (RS) realiza esto teniendo en cuenta distintos percentiles encontrados en los datos. El resultado de esto es que el SS es influenciado más fuertemente por datos lejos del promedio, cosa que el RS no sufre por la discriminación de percentiles que realiza.

Metodológicamente es un error entrenar y validar el comportamiento de una red neuronal con el mismo conjunto de datos, ya que es predecible un éxito del 100%. Por esto, con los

datos pre-procesados se separan los datos en conjuntos para entrenar y validar el modelo. De las 25 muestras para cada tipo de clasificación (Front y Back) se destinaron 70% para el entrenamiento de la red neuronal, y el 30% restante para su posterior validación utilizando la función *test_train_split* de las librerías de sci-kit learn.

La red neuronal con la que se trabajó es un clasificador MLP (Multi-layer Perception Classifier), inicializado utilizando funciones de las librerías importadas. El clasificador luego es entrenado con los datos designados para el entrenamiento. Igualmente, se utilizaron dos clasificadores MLP, uno entrenado con los datos provenientes de la normalización *Standard* y otro entrenado con los datos provenientes de la normalización *Robust*. Durante el procedimiento se mostraron los resultados y predicciones de cada etapa de entrenamiento. Entre la información mostrada están las matrices de confusión, las cuales explican la forma en la que se realizó el entrenamiento de la red neuronal y como fueron clasificadas las referencias. La matriz de confusión es una herramienta sencilla pero útil para saber en qué se equivoca el sistema cuando está clasificando, en caso de que ocurran fallas.

Finalmente, en el código se realizó una prueba con la función *cross_val_score*, donde se verifica con información concreta el desempeño de los clasificadores. La función ayuda a realizar validaciones del entrenamiento separando los datos en distintos grupos y probando cada uno individualmente. Los archivos de datos pre-procesados y las redes neuronales entrenadas son guardados en la carpeta, para que puedan ser utilizadas en la puesta a prueba in situ de los algoritmos. Los resultados del entrenamiento, las matrices de confusión, las predicciones del sistema y la validación cruzada (cross validation con *cross_val_score*) se pueden ver en la Ilustración 15, Ilustración 16 e Ilustración 17. Como se puede ver, ambos modelos tienen puntuaciones de entrenamiento promedio superiores a 95% de precisión en la toma de decisiones después de experimentar 10 iteraciones de pruebas. El promedio de precisión para los datos separados para hacer las pruebas es de 93.5% para SS y 87% para RS. El modelo realizado con el pre-procesamiento de los datos de *StandardScaler* finalmente aparentó tener mejor desempeño que el otro, por lo que opta por utilizar este para el resto del proceso del trabajo. Una certeza del 90% suficientemente buena para tener respuestas confiables en el proceso.

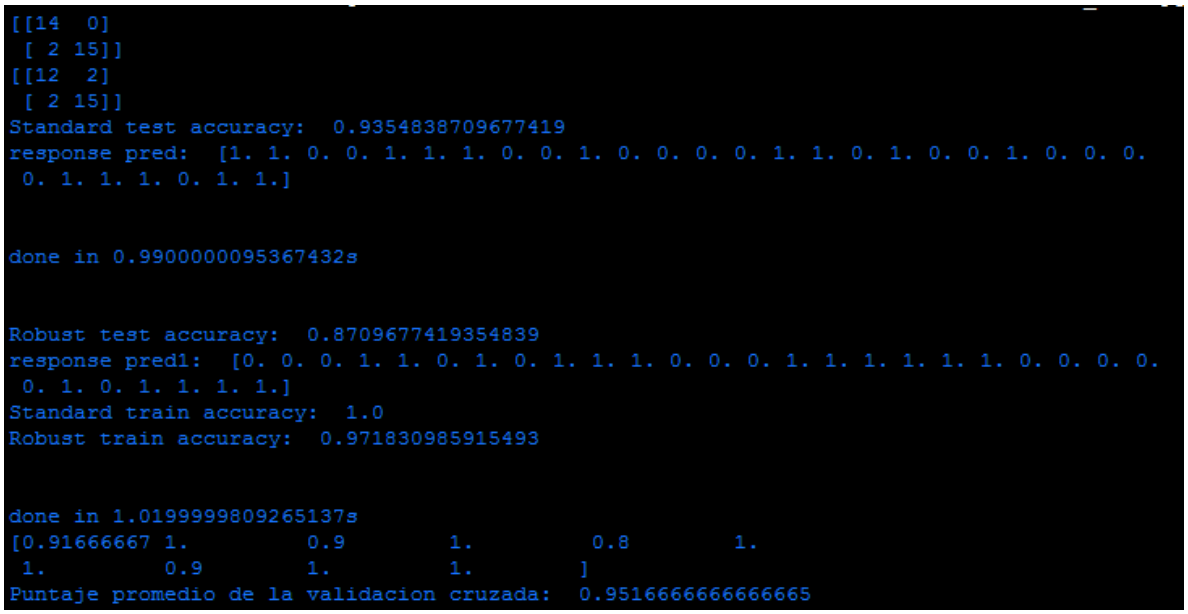

Ilustración 15: Respuesta del entrenamiento, validación y predicción de las dos redes neuronales para Referencia 1

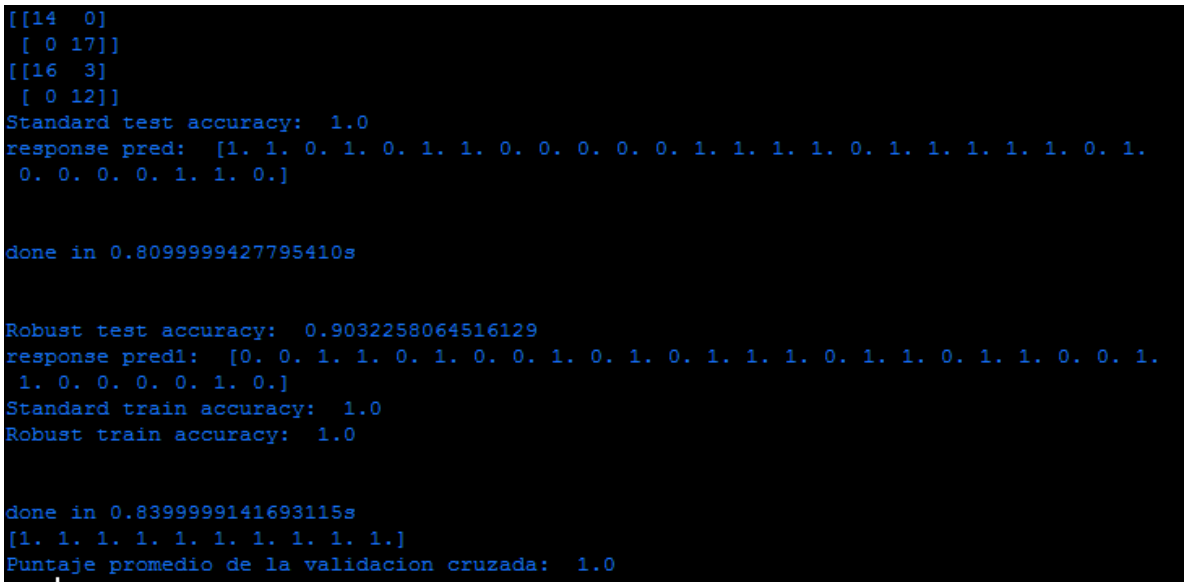

Ilustración 16: Respuesta del entrenamiento, validación y predicción de las dos redes neuronales para Referencia 2

```
[[16  0]
 [ 0 15]]
[[17  0]
 [ 0 14]]
Standard test accuracy:  1.0
response pred:  [1. 0. 1. 0. 1. 0. 1. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1.
 1. 1. 1. 0. 0. 1. 1.]


done in 0.5900001525878906s


Robust test accuracy:  1.0
response pred1:  [1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 1. 1. 1. 0. 1. 1. 1.]
Standard train accuracy:  1.0
Robust train accuracy:  1.0


done in 0.6200001239776611s
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Puntaje promedio de la validacion cruzada:  1.0
```

Ilustración 17: Respuesta del entrenamiento, validación y predicción de las dos redes neuronales para Referencia 3

El mismo procedimiento realizado desde el literal 3.2 es repetido para las otras 2 referencias de botones. El procedimiento es casi idéntico, exceptuando los umbrales y la forma de binarización de las imágenes. De igual forma los mejores resultados en general fueron vistos con el escalador de datos robusto, o Robust Scaler por encima del Standard Scaler. Las mismas características de los contornos encontrados fueron utilizadas para realizar el entrenamiento de los clasificadores. La principal diferencia entre la primera referencia y las otras dos con la que se realizó el trabajo es que las piezas de las últimas referencias son oscuras, lo que resulta en una inversión de la forma de binarizar las imágenes. Estos datos son resultado de la simulación, y por lo tanto están sesgados frente a un entorno real, ya que en un entorno real y en vivo las condiciones pueden variar. Es por esto que después de realizar el procesamiento y entrenamiento de las otras dos referencias se continuó con la validación in-situ de los modelos planteados.

3.2.5 Programación para la validación en vivo de los sistemas entrenados

Para realizar la prueba en vivo de los algoritmos ya desarrollados fue necesario escribir un script nuevo de Python. La lógica de la programación del programa es la que sigue: El programa debe prender la cámara y comenzar a detectar objetos y sus contornos en el área de trabajo. Cuando el sistema vea que no existe movimiento, o que la maquina realiza una pausa, se le extraen las características para la identificación de piezas al contorno con mayor área encontrado. Luego de extraer sus características se cargan los archivos en los que están guardados los clasificadores entrenados para determinar la posición correcta o incorrecta de la pieza de una determinada referencia. Finalmente, el programa realiza una predicción si el botón que está analizando esta al derecho o al revés imprimiendo un 0 o un 1, respectivamente. Después de la identificación, el sistema de remoción de la pieza debe tomar acción, por lo que el programa debe esperar al menos medio segundo antes de continuar examinando las siguientes piezas. El programa se

puede detallar en el Código 8, y un diagrama de flujo de cómo funciona en la Ilustración 18.

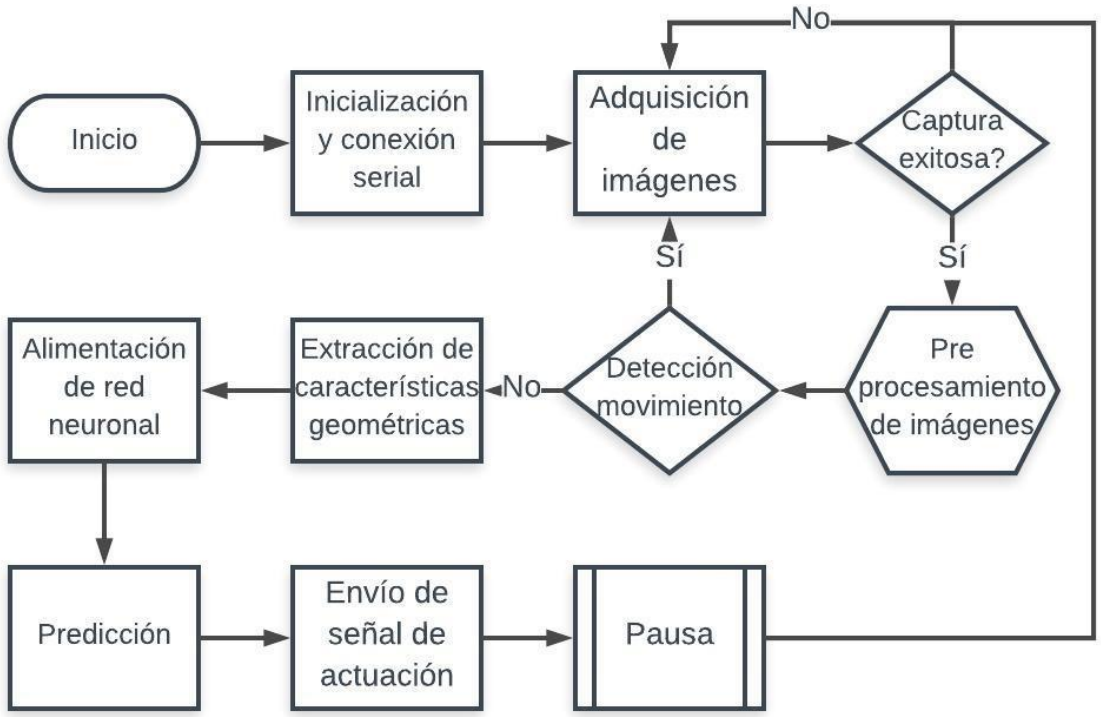

Ilustración 18: Diagrama de flujo de algoritmo de detección en vivo

De igual forma se realizaron otros dos programas casi idénticos para las otras dos referencias de piezas, cada uno con sus cambios relevantes en cuanto al umbral y la forma de binarización. En esta etapa es importante mantener los mismos valores de pre procesamiento para que el sistema pueda identificar correctamente cada pieza. Para que el programa sepa cuando tomar la foto, o cuando identificar que la maquina está en una pausa, se le calcula el centro del contorno con mayor área en sentido vertical, y si su valor es igual en dos frames consecutivos significa que la imagen ya no tiene movimiento. Además de esto, el programa se asegura que el centro del contorno que está procesando este en un margen de coordenadas, para tener aún más certeza que el objeto en cuestión sea el correcto y que se encuentre en a posición correcta.

```
# -*- coding: cp1252 -*-
import numpy as np
import cv2
from glob import glob
from sklearn.externals import joblib
import serial
import time
import xlrd

from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler, RobustScaler

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.externals import joblib
from sklearn.model_selection import cross_val_score


captura = cv2.VideoCapture(0)
time.sleep(2)
thresh = 170
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
y_anterior = 100000

while(captura.isOpened()):
    ret, frame = captura.read()

    if ret == True:

        #print 'processing %s...' % fn

        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        imgBilateral = cv2.bilateralFilter(gray,9,75,75)

        h,w = imgBilateral.shape[:2]
        imgBilateral[:h,:205] = 0


        ret,edges = cv2.threshold(imgBilateral,thresh,255,cv2.THRESH_BINARY)
        erosion = cv2.erode(edges,kernel=kernel,iterations = 1)
```

```
cv2.imshow("edges", erosion)
cv2.waitKey(1)

drawing = np.zeros(frame.shape,np.uint8)     # Image to draw the contours

# Encontrar contornos

contours,hierarchy = cv2.findContours(erosion.copy(),
                                      cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

try:

    cnt = max(contours, key=cv2.contourArea)

    x,y,w,h = cv2.boundingRect(cnt)

    if((y+h/2) == y_anterior):
        captura.release()
        captura = cv2.VideoCapture(0)
        ret, frame = captura.read()
        time.sleep(.1)
        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        print(cv2.contourArea(max(contours,key=cv2.contourArea)))

        imgBilateral = cv2.bilateralFilter(gray,9,75,75)
        h,w = imgBilateral.shape[:2]
        imgBilateral[:h,:205] = 0

        ret,edges1 = cv2.threshold(imgBilateral,
                                           thresh,255,cv2.THRESH_BINARY)
        erode = cv2.erode(edges1,kernel=kernel,iterations = 2)

        contours,hierarchy = cv2.findContours(erode.copy(),
                              cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

        cnt = max(contours, key=cv2.contourArea)

        x,y,w,h = cv2.boundingRect(cnt)

        print(y+h/2)
```

```
                    cv2.drawContours(drawing,[cnt],0,(255,255,255),-1)
                    cv2.rectangle(drawing,(x,y),(x+w,y+h),(0,0,255),1)
                    M = cv2.moments(cnt)
                    cx = int(M['m10']/M['m00'])
                    cy = int(M['m01']/M['m00'])
                    A = cv2.contourArea(cnt)
                    p = cv2.arcLength(cnt,True)
                    Comp = A/float(p*p)
                    RA = w/float(h)
                    Hu = cv2.HuMoments(M)
                    X = np.array([A, p, Comp, RA, Hu[0][0], Hu[1][0],
                                  Hu[2][0], Hu[3][0], Hu[4][0], Hu[5][0],
                                  Hu[6][0]], dtype = np.float32)
                    cv2.imshow("img", drawing)
                    cv2.waitKey(1)
                    if (cy >= 100 and cy <= 300):
                        modelo = joblib.load("MLP_Robust.sav")
                        robust_scaler = joblib.load("Robust.sav")
                        XX = robust_scaler.transform(X)
                        X_test = np.array(XX).reshape(1,-1)

                        prediccion = modelo.predict(X_test)
                        print prediccion[0]
                        time.sleep(0.5)
                y_anterior = (y+h/2)
            except:
                pass
        else:
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            captura.release()
            break
cv2.destroyAllWindows()
```

Código 8: Script para la validación en vivo del sistema

3.3 Implementar del dispositivo funcional en un tambor vibrador

A continuación, se explica la forma en la que se desarrolló la principal tarea de ensamble y en la que se pudo comenzar a demostrar un resultado efectivo del trabajo. Una vez tenidas las distintas partes que componen el dispositivo, desarrolladas en los capítulos

anteriores, se pudo comenzar con la implementación del sistema físico y digital completo. Es en esta fase en la que se pudo ver el conjunto de todo lo que se ha trabajado hasta el momento. Gracias a los esfuerzos realizados en la fase de diseño y fabricación, la implementación o ensamble de las piezas en la zona de trabajo debe ser de gran facilidad. Ésta fue la filosofía que se tuvo en mente a la hora de diseñar y ensamblar.

Para el ensamble de sistema primero se tuvo que limpiar el área de trabajo y desocupar a la máquina ensambladora de toda su producción por unas cuantas horas, mientras se realizaba el ensamble de las partes. Dado el diseño del mecanismo de extracción de las piezas y por esto el ligero cambio en el sistema de acople de la base para la cámara, lo primero que se debe instalar en la máquina es la platina de acople, ubicada debajo del riel de ensamble. Su diseño tuvo en consideración la forma en la que está físicamente ensamblado el riel como tal, aprovechando los tornillos presentes como se pueden ver en la Ilustración 2, haciendo que la implementación de la platina sea de gran facilidad. Montando la platina se pudo confirmar el diseño y la ejecución exitosa de la pieza, ya que los agujeros realizados y la ubicación de la boquilla para la extracción neumática de las piezas estaban posicionados correctamente en el riel.

A continuación, se ubicó la manguera neumática en la boquilla para sostenerla, y se conectó a la electroválvula, la cual a su vez estaba conectada a la red de aire comprimido existente en la empresa, la cual tiene una compresión de aproximadamente 110 psi, pero cuenta con una válvula para manualmente ajustar la presión deseada. Cómo la electroválvula escogida es normalmente cerrada, no se necesitan consideraciones adicionales en la conexión neumática del trabajo. Consiguientemente, se realizó la conexión eléctrica y electrónica de la válvula, siguiendo el diseño realizado en los capítulos anteriores. El solenoide de la válvula fue alimentado por una fuente directa de 24V, los cuales fueron switcheados por un relay comandado por el Arduino. Estas partes electrónicas fueron conectadas entre sí mediante una proto-board. La tarjeta de Arduino fue alimentada por el computador en el que se realizó el desarrollo de detección y la fuente de potencia portátil fue ubicada apropiadamente en el área de trabajo, detrás de la máquina y fuera de alcance. De esta forma se evitan tropiezos o cortocircuitos accidentales.

Finalmente, el sistema de adquisición de imágenes fue ensamblado utilizando la platina de acople y sus tornillos respectivos. Se pudo confirmar el correcto diseño y cambios realizados en el primer diseño de la base de la cámara, ya que el sistema de detección se pudo ensamblar fácilmente con la platina de acople, como se puede ver en la Ilustración 19.

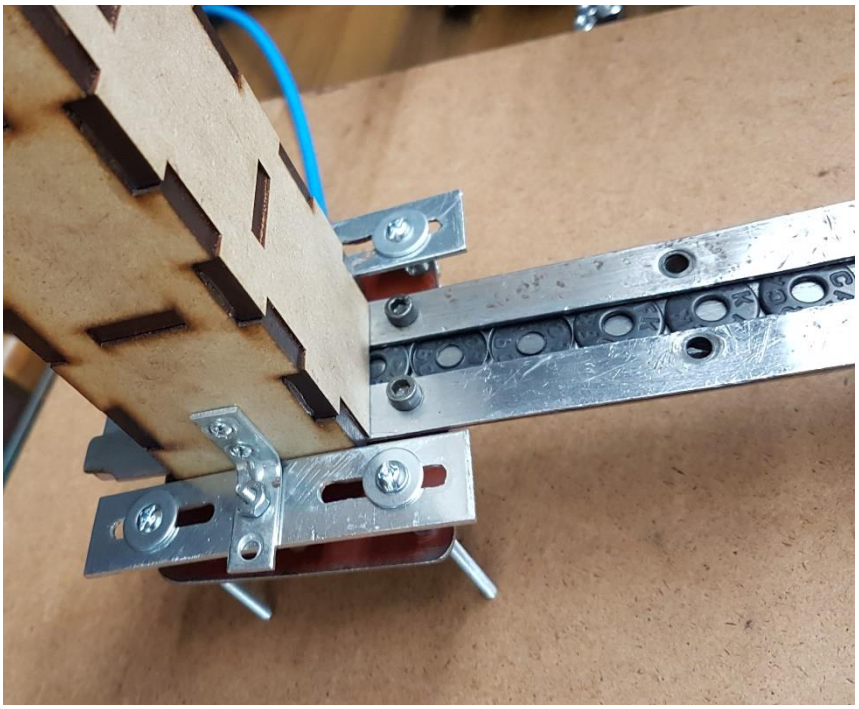

Ilustración 19: Ensamble de platina de acople, riel y sistema de adquisición

3.4 Validación en vivo del sistema de identificación y remoción de piezas en la empresa C.I. Estrada Velásquez

Durante el desarrollo de las distintas etapas del trabajo, se realizaron verificaciones individuales para confirmar el funcionamiento de cada etapa luego de ser diseñada e implementada. Algunas de estas pueden ser evidenciadas en los capítulos anteriores, donde se corrieron simulaciones y programas de validación del algoritmo. Adicionalmente, durante la fabricación de las partes del sistema, como la base para la cámara y el sistema de remoción, pequeñas verificaciones fueron realizadas en cuanto al correcto ensamble entre los componentes físicos de las estructuras.

3.4.1 Validación parcial fuera del área de trabajo

Antes de realizar la validación final del proyecto se hizo una parcial, la cual se pudo realizar de forma más rápida y fácil que la definitiva. Se desmonto el riel de ensamble de la máquina para facilitar el entorno de trabajo, y se acoplo el sistema de detección. Los sistemas eléctricos y electrónicos también fueron conectados, pero sin su parte neumática, ya que el objetivo de la prueba era validar las capacidades de detección y comunicación entre el sistema digital y el mecánico.

En la prueba, manualmente se introdujeron distintas pizas de las diferentes referencias utilizadas a lo largo de la realización del trabajo por el riel de ensamble desmontado. Con

la unidad de procesamiento encendida y trabajando, la electroválvula conectada a su potencia y señales de actuación, y la tarjeta de Arduino recibiendo órdenes del computador se verificó el funcionamiento del sistema de detección. Mientras las detecciones e identificaciones de las piezas eran realizadas, pudo verse también la reacción de la electroválvula gracias al testigo, o LED que tiene internamente. Evidencia de estas pruebas puede ser vista en los videos anexos al presente trabajo, y en la Ilustración 20.

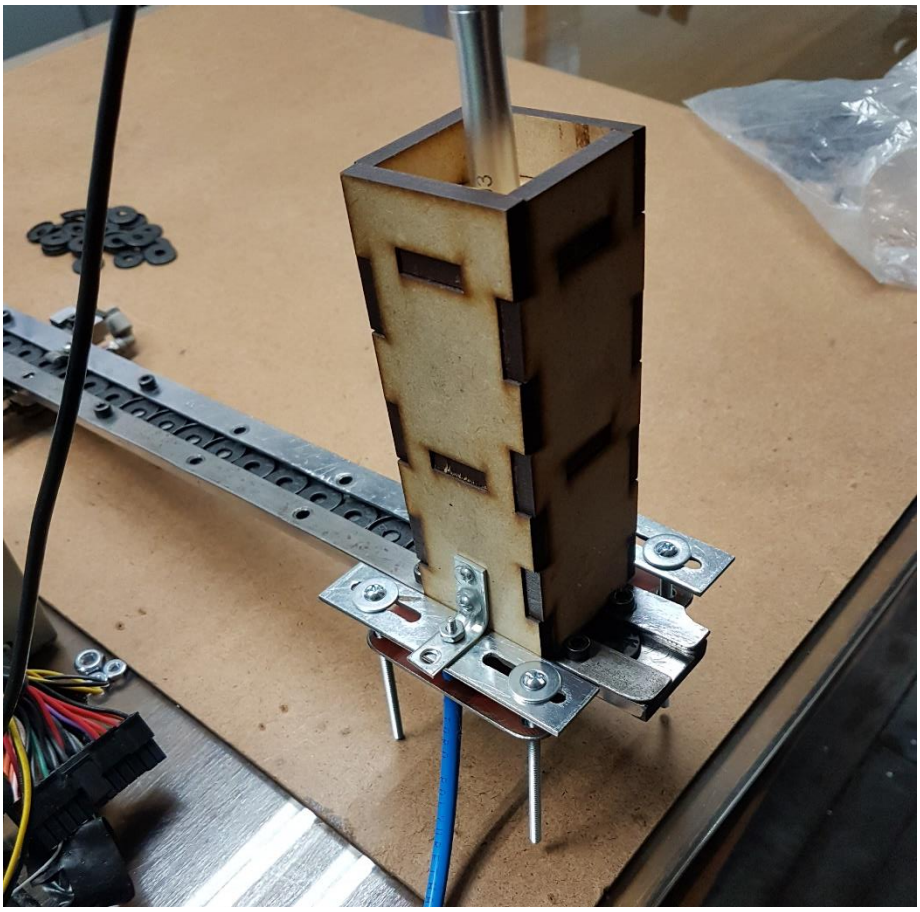

Ilustración 20: Prueba en vivo del sistema de detección

Es en este punto donde se pudo verificar principalmente la eficacia de los algoritmos desarrollados en los capítulos anteriores. Para hacerlo se realizó un conteo de las veces que el programa se equivocó, o actuó de forma incorrecta o inesperada con cada referencia de botón. Este conteo se puede expresar como un porcentaje de efectividad de los algoritmos de detección.

El resultado del conteo mostro que para las últimas dos referencias, el sistema fue capaz de identificar 94.6% de las piezas (90 de 95 muestras) que se encontraban en una posición incorrecta para la referencia número 2, y 92.4% de las piezas (84 de 91 muestras) que se encontraban al revés para la referencia número 3. Estas cifras son alentadoras, ya que son congruentes con las distintas simulaciones realizadas en los capítulos anteriores, y demuestran que los algoritmos de detección e identificación para estas dos referencias fueron exitosos. Ejemplos de las imágenes con las que el sistema tomo las decisiones de separación se pueden ver en la Ilustración 21.

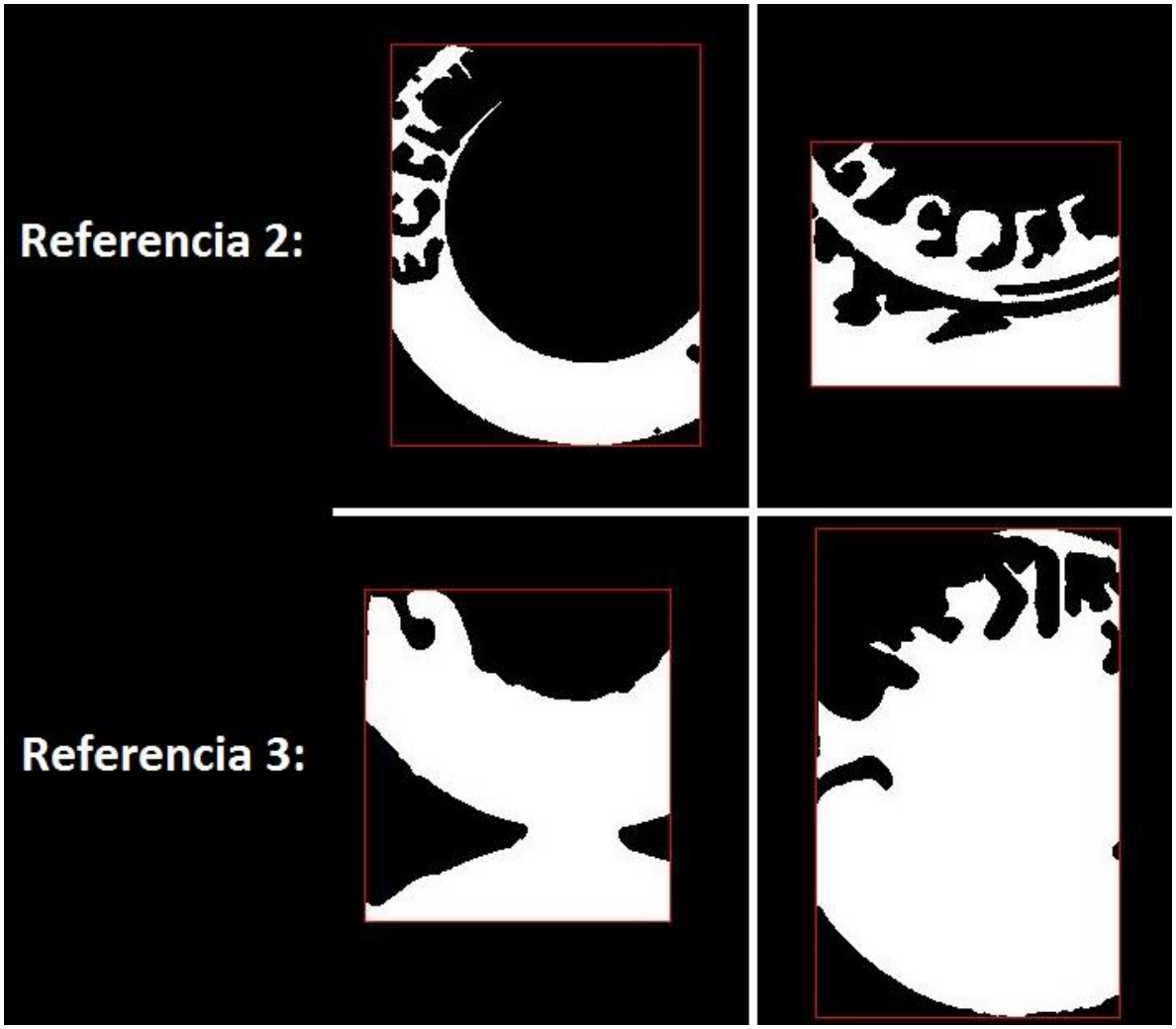

Ilustración 21: Contornos analizados de las referencias 2 y 3

Sin embargo, para la referencia número 1, el porcentaje de identificación exitosa fue de 61.3% (60 de 98 muestras). Para ver lo que ocurría con esta referencia en particular, se indagó más a fondo en la forma de tomar decisiones del sistema, y se analizaron las imágenes que el algoritmo estaba procesando. Ejemplos las imágenes pueden ser vistas en la Ilustración 22.

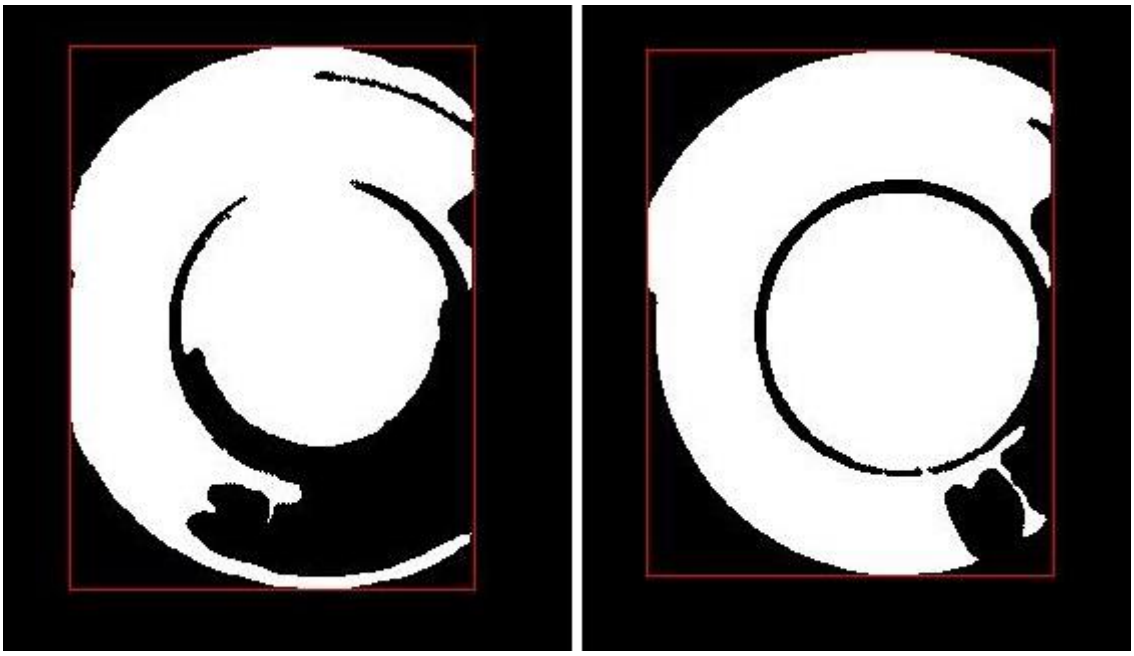

Ilustración 22: Contornos analizados de la referencia 1

Como se puede ver, la semejanza de las imágenes es demasiada para que el sistema sea capaz de diferenciarlas. Al ver las imágenes originales de las piezas en la Ilustración 23, se puede apreciar más a fondo la situación. Estos resultados son evidencia que la identificación por medio de contornos no es apropiada para la referencia 1 en las condiciones actuales de instrumentación e iluminación.

Ilustración 23: Imágenes originales de la referencia 1 al derecho y al revés

Asumiendo que el problema puede llegar a ser de *underfitting* se procede a entrenar el sistema con más muestras, esta vez de 100 imágenes por cada lado del botón, o sea 200 para esta referencia, y validar de nuevo el sistema de detección. Para esto se utilizan los códigos previamente desarrollados mostrados en el Código 2, el Código 6, el Código 7 y el Código 8. Además de esto se utilizaron distintos métodos de clasificación, como el *K-Nearest Neighbors* (KNN) y *Support Vector Classification* (SVC). Los cambios en particular se pueden ver en el Código 9.

Luego de esto se procede a verificar nuevamente la identificación de las piezas. Aunque la validación por medio de simulación a la hora de realizar el entrenamiento muestra buenos resultados, al hacer la prueba física fuera del área de trabajo se pudo confirmar que la identificación no mejora, ya que se obtienen resultados similares a los resultados obtenidos anteriormente. Ambos lados del botón son demasiado parecidos como para realizar una identificación por factores geométricos, y debido a esto se decidió dejar de continuar con esta referencia para su remoción. Para desarrollar la identificación de piezas como esta, es necesario utilizar métodos diferentes a la extracción de características geométricas.

3.4.2 Validación final en el área de trabajo

La validación final del sistema fue realizada en el área de trabajo, donde opera normalmente la máquina de ensamble. Para la realización de las pruebas de funcionamiento del sistema de detección no es necesario que la maquina realice el ensamble como tal del botón, sino que se mueva para permitir la identificación de una de las piezas que lo conforman. Vale la pena mencionar, además, que la velocidad con la

que puede funcionar la maquina es variable mediante un potenciómetro ya instalado. Esta funcionalidad resulta sumamente práctica para realizar las pruebas de funcionamiento.

En cuanto a la identificación digital de las piezas, los resultados de esta validación deben de ser similares a los resultados de la prueba realizada anteriormente, ya que el sistema de detección permanece siendo el mismo, y el propósito del diseño del mismo es precisamente que las condiciones visuales para realizar la identificación no cambien por el entorno. El propósito principal de la validación final es evaluar el comportamiento físico de remoción de piezas en el área de trabajo.

3.4.2.1 Prueba de remoción de piezas sin sistema de detección

Inicialmente el ensamble fue realizado siguiendo los diseños de las distintas partes que conforman el sistema de remoción, y asegurándose que las conexiones entre los sistemas sean correctas, de igual forma que se hizo la validación fuera del área de trabajo, como se discutió anteriormente. De esta forma se puede validar la dinámica de la separación de piezas antes que el sistema de detección esté ensamblado. El único cambio en cuanto a las conexiones es que la actuación de la válvula se realiza manualmente, para controlar y probar la remoción de las piezas. Parte de la prueba se puede ver en uno de los videos anexos al presente trabajo.

Para que el sistema de remoción funcione correctamente se deben cumplir dos requisitos: Primero, al accionar la válvula se debe remover la pieza objetivo, es decir, la que está siendo identificada por el sistema de adquisición visual. Segundo, se debe extraer únicamente la pieza a la que se le está realizando la identificación. La remoción de la misma no debe afectar a las piezas que siguen en la línea de ensamble. El sistema de remoción fue diseñado específicamente para poder cumplir con estos requisitos.

En la prueba de la remoción de piezas se pudo ver el comportamiento de las mismas con el aire comprimido utilizado. El primer requisito para que sea efectiva la remoción siempre se cumple, la pieza objetivo siempre logra ser removida. El segundo requisito no siempre se cumple, ya que algunas veces la separación de una pieza hace que la siguiente en la línea caiga a su posición y escape por el camino por donde fue removida la pieza anterior. Este comportamiento baja la eficiencia del dispositivo de identificación y remoción, y puede afectar su posterior funcionamiento.

El diseño de la remoción de piezas tenía pensado evitar este comportamiento haciendo que las piezas siguientes en la línea de ensamble flotaran en el colchón de aire generado por el aire bajo ellas, pero no se contó con el peso y el empuje que tienen las piezas provenientes del tambor vibrador. Este empuje adicional hace que cuando las piezas caigan a su posición de identificación, la inercia que traen, la fuerza generada por el peso y vibración de la piezas encima, y la geometría de la zona en donde llega la pieza hace que se escapen lateralmente por el riel, a través del agujero hecho para remover las piezas del ensamble.

Para corregir el comportamiento equivocado con el sistema de remoción se le agrega un obstáculo a la vía por donde son separadas las piezas. El obstáculo debe impedir el paso

de piezas lo suficiente para que el peso y movimiento de la misma no sea suficiente para pasarlo, pero no tanto como para que la acción de remoción de aire comprimido no sea suficiente para separar las piezas por el camino. La compuerta diseñada para realizar esto mismo demostró ser ineficiente para lograr este objetivo, por lo que es necesario optar por métodos alternos. Se probó con aumentar el nivel o distancia desde la superficie del riel de la zona por donde escapan las piezas para que no continúen deslizándose indebidamente. Para lograr esto se utilizó una platina metálica de 0.3mm de espesor. La idea de esta modificación se puede ver en la Ilustración 24.

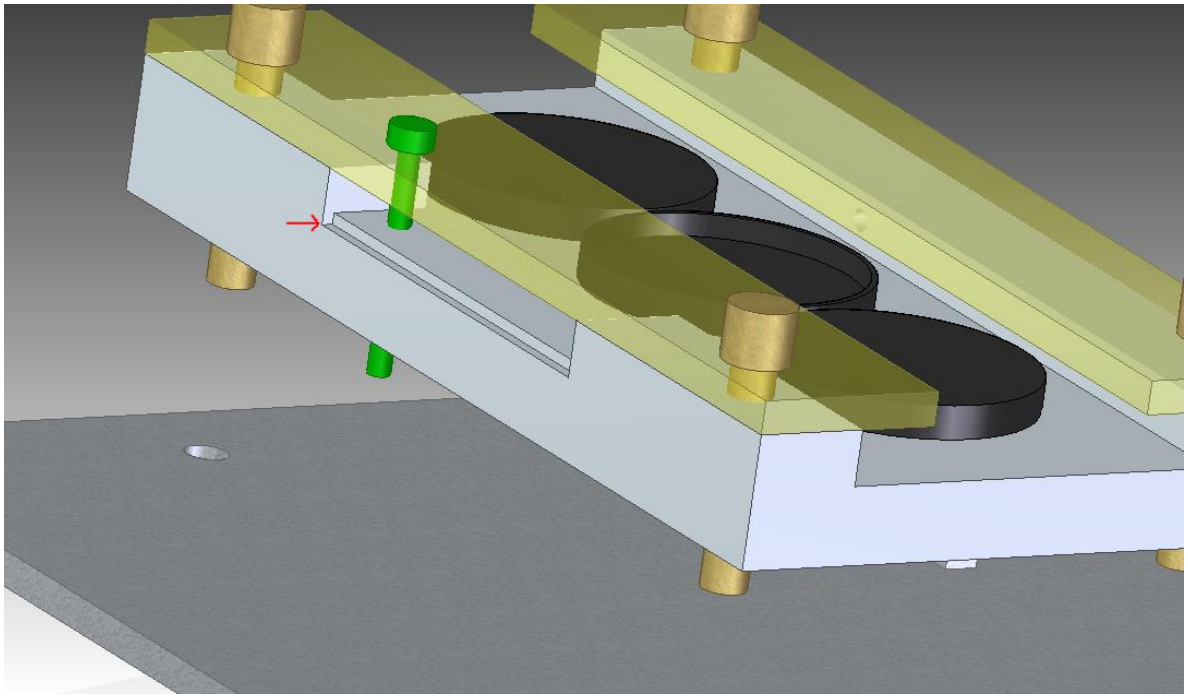

Ilustración 24: Adición de andén para el riel

Al probar de nuevo el sistema se logró comprobar que el cambio realizado fue efectivo para evitar el deslizamiento y el escape de piezas que no deben de salir de la línea de ensamble. El aumento en nivel de la superficie del riel actúa como andén para las piezas que caen después de que una es removida.

3.4.2.2 Prueba del sistema completo de identificación y remoción de piezas

Después de realizar las pruebas del sistema de remoción, se continuó con ensamblar el sistema de detección para realizar la validación del dispositivo completo. Los criterios para evaluar el sistema es el porcentaje de piezas que se encuentran al revés que el sistema es capaz de separar exitosamente, dada una muestra de alrededor de 100 piezas. Este procedimiento se repite para cada referencia. La Ilustración 25 muestra el ensamble en el área de trabajo.

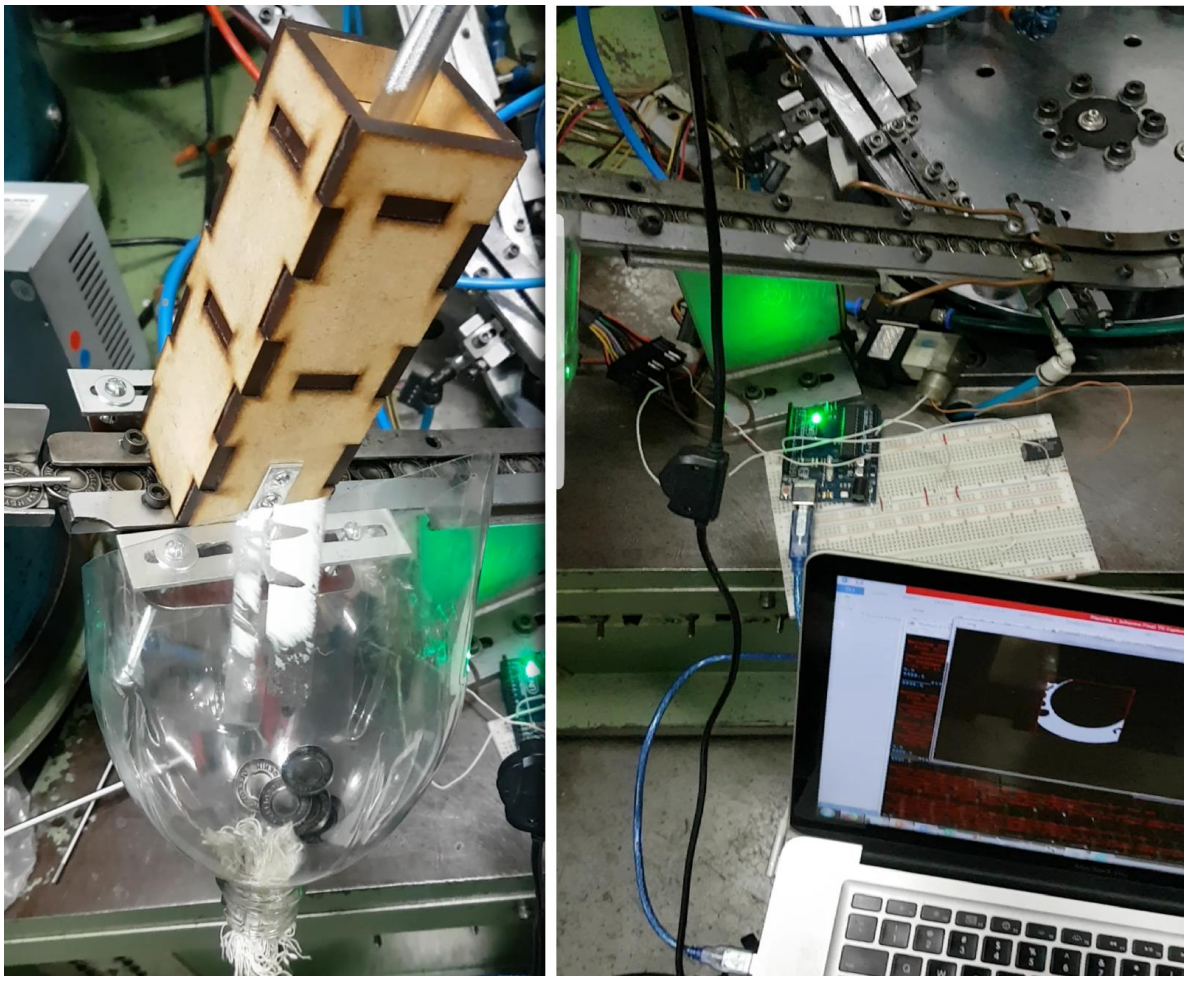

Ilustración 25: Ensamble completo en zona de trabajo

Se inició la prueba llenando la tolva o el tambor vibrador de piezas para ser separadas, inicialmente de la referencia 2. Las piezas automáticamente suben por el tambor vibrador hasta el riel, y se llenó el riel de piezas para poder comenzar. Algunas de las piezas en la línea de ensamble son puestas manualmente al revés para agilizar el procedimiento. Se comienza por accionar la maquina manualmente para tener controlado el tiempo que se demora el sistema en detectar las piezas, ya que en la programación se tienen varias pausas para garantizar el correcto funcionamiento del sistema. Se pudo ver que el desempeño del sistema de adquisición y el de remoción actúan de forma correcta.

Para realizar la validación con la maquina ensambladora moviéndose automáticamente se configuraron los tiempos de pausas en los algoritmos de detección y remoción para que se ajuste con la velocidad a la que se opera la máquina. Para realizar la prueba, se ajusta la máquina para que haga un ensamble por segundo, lo que significa que la suma de todos los retrasos en el código no puede superar un segundo. Se modificaron los tiempos en los algoritmos, y se comenzó con la prueba del sistema en conjunto. Estos cambios

son fáciles de hacer gracias a que el computador en donde corre el algoritmo se encentra en la zona de trabajo, y se pueden hacer cambios y ver el rendimiento del sistema de detección en vivo.

Inicialmente se pudo notar que un ensamble cada segundo resulta ser muy rápido para la cámara utilizada, ya que en algunas ocasiones se pudo presenciar errores de reconocimiento. Los errores de reconocimiento se ven en el computador en donde corre el programa. Dada la alta velocidad de ensamble, la cámara utilizada no siempre logra enfocar y capturar la pieza después de que otra es extraída, generando problemas en la captura y por lo tanto en el reconocimiento y extracción. Después de realizar pruebas e intentar diferentes tiempos con el sistema de remoción y la maquina ensambladora, se cambiaron los tiempos de ensamble y de retrasos de los algoritmos para tener un ensamble cada 1.5 segundos. Este tiempo depende de la dinámica con la que se mueven las piezas y la velocidad de captura que tiene la cámara. Con este cambio se pudo ver en la ejecución en vivo del programa que las piezas para botones fueron identificadas correctamente.

Luego del cambio de velocidades, el sistema comenzó a identificar erróneamente las piezas. Los porcentajes de certeza en vivo no estaban cerca a los que se pudieron presenciar en las simulaciones y en la prueba física por fuera del área de trabajo. Para identificar el problema se analizaron las imágenes que estaban siendo sometidas al análisis del algoritmo de detección. Luego de inspecciones detalladas de las distintas imágenes se pudo finalmente descubrir la razón de las diferencias de operación y rendimiento del sistema. La base de la cámara estaba casi un milímetro desfasado de su posición ideal. Este pequeño cambio de posición hace que el pre procesamiento de las imágenes cambie, y que el contorno analizado sea distinto a los que utilizó el programa para entrenar la red neuronal. Un ejemplo del cambio se ve en la Ilustración 26. Aunque la diferencia es poca, es representativa a la hora de extraer características geométricas. Fue de gran importancia descubrir esto, ya que implica que el entrenamiento de la red neuronal no recibía pequeños cambios de entorno, haciendo que el sistema sea más propenso a fallar, y menos robusto.

Ilustración 26: Efecto de desfase de piezas en el riel

Este tipo de error es similar a los errores que son resultado de un *overfitting,* en donde el sistema es entrenado con demasiadas muestras con características similares. Debido a la forma en la que están ubicadas las piezas en el riel de ensamble, pequeños movimientos de las piezas dentro de este hacen que la adquisición de las imágenes y sus características geométricas cambien representativamente, porque las tapas en el riel de ensamble ocultan pedazos de las piezas. Al ocurrir esto, en la adquisición e las imágenes solo se obtiene una fracción de la pieza con el que se entrenó el sistema. Para corregir el problema se decidió probar cambiar el clasificador utilizado para entrenar la red neuronal. El que se utilizaba anteriormente era MLP, se debe devolver al código y reentrenar el sistema con clasificadores KNN o SVC, que son de implementación rápida y generalmente entregan buenos resultados. Específicamente el método de KNN es no paramétrico, y resulta ser flexible para casos en donde las entradas de los datos cambian moderadamente. Los cambios se deben realizar en el algoritmo de entrenamiento del Código 7. La gran mayoría del código fue el mismo, los principales cambios son el tipo de clasificador utilizado. El cambio en los algoritmos se puede ver en el Código 9.

```
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier

svm = SVC(C=10, kernel="linear")
svm.fit(samples_train,responses_train)

knn = KNeighborsClassifier(n_neighbors=5, weights='distance', n_jobs=1)
knn.fit(samples_train, responses_train)
```

Código 9: Cambio de clasificadores en el algoritmo de entrenamiento.

Las etapas de simulación y validación digital fueron realizadas para cada uno de los clasificadores utilizados. Ambos demostraron tener muy buenos resultados, y ambos mostrando resultados de porcentajes por encima del 94% para cada referencia. Sin embargo, se decidió continuar con el método de K Vecinos más Cercanos (KNN por sus siglas en ingles), ya que el método es no paramétrico y generalmente devuelve buenos resultados cuando la barrera de decisión es irregular (Vanderplas, 2011). Los resultados de entrenamiento y validación se pueden ver en la Ilustración 27, la Ilustración 28 y la Ilustración 29. Aunque cada entrenamiento parece ser efectivo, el desempeño de los modelos en las pruebas en vivo es diferente a lo que muestran las simulaciones.

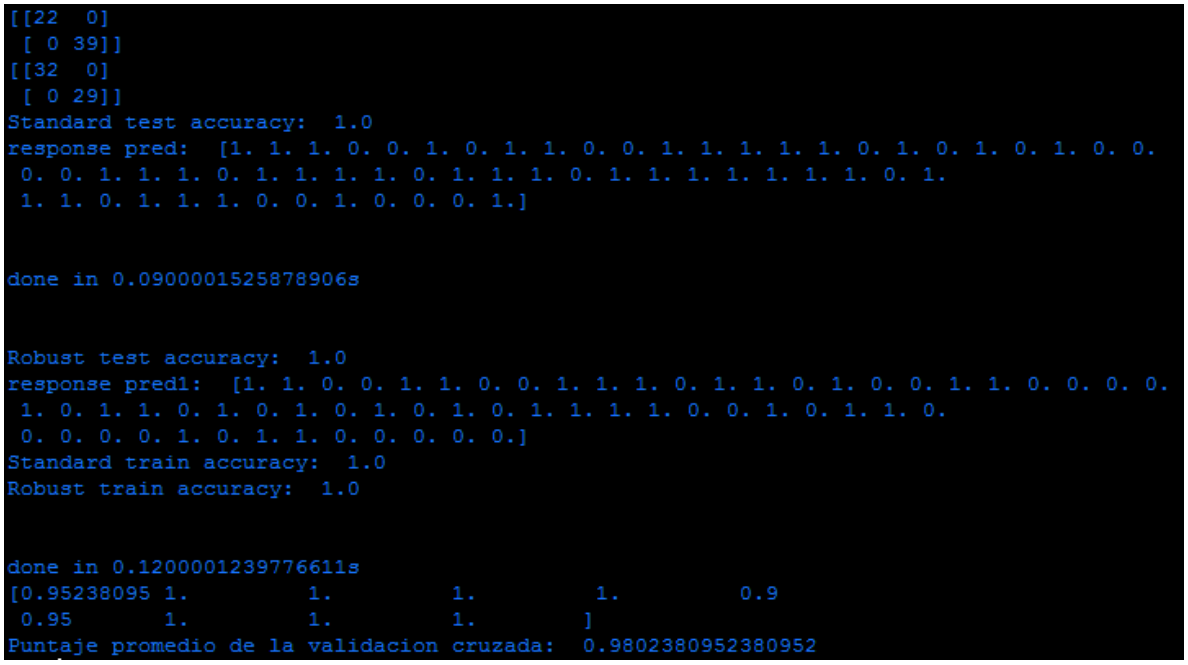

Ilustración 27: Resultados de entrenamiento y validación KNN para Referencia 2

```
[[28  0]
 [ 2 31]]
[[32  1]
 [ 0 28]]
Standard test accuracy:  0.9672131147540983
response pred:  [0. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0. 1.
 1. 0. 0. 1. 0. 1. 1. 0. 0. 1. 0. 1. 1.]


done in 0.2400000095367432s


Robust test accuracy:  0.9836065573770492
response pred1:  [0. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0. 1. 0. 1. 1. 1.
 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 0.
 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0.]
Standard train accuracy:  0.9785714285714285
Robust train accuracy:  0.9428571428571428


done in 0.2699999809265137s
[0.85714286 1.         1.         0.95       1.         0.85
 0.95       0.85       1.         0.95      ]
Puntaje promedio de la validacion cruzada:  0.9407142857142856
```

Ilustración 28: Resultados de entrenamiento y validación con MLP para Referencia 2

```
[[38  0]
 [ 0 23]]
[[31  0]
 [ 0 30]]
Standard test accuracy:  1.0
response pred:  [0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 1. 1. 1. 1. 0. 0. 0. 1. 1.]


done in 0.0699999332427979s


Robust test accuracy:  1.0
response pred1:  [1. 0. 0. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 1. 1. 1.
 0. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 1. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1.
 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 0. 0.]
Standard train accuracy:  1.0
Robust train accuracy:  1.0


done in 0.09999990463256836s
[1.   1.   1.   1.   1.   0.95 1.   1.   1.   1.  ]
Puntaje promedio de la validacion cruzada:  0.9949999999999999
```

Ilustración 29: Resultados de entrenamiento y validación con SVC para Referencia 2

Con los cambios realizados en la programación se pudo continuar con la validación en vivo del sistema. El funcionamiento del sistema se puede ver en los videos anexos. Para cada referencia se realizó un conteo de veces que el sistema se comportó correctamente dependiendo de la orientación de la pieza. Para la referencia 2 se tuvo un 90.4% de

certeza (85 de 94 muestras), y para la referencia 3 un 91.7% de certeza (89 de 97 muestras), en la Tabla 4 se pueden ver estos datos. La referencia 1 demostró ser inadecuada para realizar la identificación por métodos geométricos, ya que los resultados fueron de alrededor de 60% de certeza.

	Referencia 2	Referencia 3
MLP	72 (76.6%)	78 (80.4%)
KNN	85 (90.4%)	89 (91.7%)
Número de muestras	94	97

Tabla 4: Resultados entre modelos de entrenamiento en vivo

4 CONCLUSIONES Y CONSIDERACIONES FINALES

Durante el diseño y desarrollo de los sistemas de identificación y de remoción de las piezas, y luego del ensamble y validación del dispositivo se puede concluir que el diseño mecánico de los sistemas es uno de los aspectos más importantes para el correcto desempeño de los mismos. Un buen diseño mecánico facilita la implementación, las pruebas y el desempeño del sistema que se está desarrollando. En el caso particular del trabajo, se pudo ver que gracias a un diseño pensado en ser simple y práctico para el montaje y desmontaje se ahorraron tiempos de implementación y se pudo trabajar de una forma flexible, permitiendo hacer cambios cuando fuese necesario.

Realizar diferentes pruebas progresivas durante el desarrollo del sistema fue de gran ayuda para evitar problemas en la operación. Estas permiten aislar y probar por partes los distintos componentes que conforman el sistema de detección y remoción. En cuanto a software, realizar pruebas en cada etapa de los algoritmos es útil para simplificar el *debugging* y para identificar posibles problemas adicionales de lógica. Físicamente, realizar los distintos ensambles por pasos ayudo a identificar problemas en la operación del sistema de remoción, cosa que pudo haber sido complejo de identificar si se hubiese validado el sistema completo.

Además, se pudo identificar en las pruebas que el algoritmo necesitaba cambio de entrenamiento, ya que con el entrenamiento inicial de MLP tuvo un peor funcionamiento que con las nuevas redes neuronales utilizadas. Esta afirmación se pudo obtener gracias a que las distintas partes que conforman el resto del sistema ya habían sido probadas, y habían mostrado comportamientos adecuados. La corrección del problema ayudo a que el sistema fuera más robusto de lo que se pensaba que era en un principio, y en la identificación de distintos métodos para realizar entrenamientos de Machine Learning.

Como se pudo ver, el método inicial de entrenamiento, MLP, aunque demostró buenos en las etapas de simulación y pruebas iniciales, finalmente mostró que el método no era lo suficientemente flexible como para adaptarse al entorno en el que trabaja el sistema de detección y remoción, y que los pequeños cambios en el ambiente de trabajo afectan importantemente el desempeño del entrenamiento realizado con el método. KNN en cambio, aunque puede resultar menos atractivo en un problema avanzado de Machine Learning donde sea necesario parametrizar y controlar minuciosamente las variables de entrenamiento, demostró que es un método de clasificación con mucha flexibilidad en cuanto a la selección de muestras basado en características marginalmente distintas a las utilizadas para el entrenamiento.

La utilización de características geométricas para realizar la diferenciación entre piezas fue parcialmente exitosa, ya que para la referencia 1 de las piezas, en donde la similitud entre el derecho y el revés de la pieza es demasiada, el pre procesamiento de las imágenes no es suficiente para encontrar diferenciadores en su geometría. Para

ocasiones como esta puede ser necesario aplicar métodos de OCR (Reconocimiento óptico de caracteres), o demás técnicas que no dependan de la forma general del objeto.

La naturaleza del entorno en el que se trabajó le agrego dificultad a las tareas de pre procesamiento e identificación de las piezas, ya que el riel de ensamble de la maquina no permite ver por completo la pieza a identificar. Esto limita un poco la adquisición de características visuales para hacer la clasificación de referencias, y obliga a que se ingenie un método no invasivo para remover las piezas. Vale la pena considerar ser más invasivo en los cambios del riel de ensamble y el área de trabajo si se desea tener un sistema más confiable. Para mejorar el desempeño del algoritmo es importante hacer que la pieza se pueda ver en su totalidad, y que el sistema de remoción sea robusto.

REFERENCIAS

Bourg, W. M., Bresnahan, S. A., Haarsma, G. J., MacGregor, J. F., Martin, P. A., & Yu, H. (2006). *Estados Unidos Patente nº US70688117B2.*

Brosnan, T., & Sun, D.-W. (2004). Improving quality inspection of food products by computer vision - a review. *Journal of Food Engineering, 61*, 3-16.

Cantero, A. D., & Martinez, E. A. (2013). Vision por computadora: identificacion, clasificacion y seguimiento de objetos. Facultad Politecnica Universidad Nacional del Este.

El Naqa, I., & Murphy, M. J. (2015). What Is Machine Learning? *Machin Learning in Radiation Oncology*, pp. 3-11.

Enrique Sucar, L., & Giovani Gomez, M. (2011). *Vision Computacional.* Recuperado a partir de http://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf.

González Marcos, A., Martinez de Pisón Ascacíar, F. J., Pernía Espinoza, A. V., Alba Elías, F., Castejón Limas, M., Ordieres Meré, J. B., & Vergara González, E. P. (2006). *Técnicas y Algoritmos Básicos de Visión Artificial.* Recuperado a partir de https://dialnet.unirioja.es/servlet/libro?codigo=338314&iframe=true&width=80%&height=80%.

Guindos Rojas, F., Piedra Fernández, J. A., & Peralta López, M. (2001). *Visión artificial con IMtdi.*

JasVisio. (2009). *Jasvisio*. Recuperado el 2017, de Aplicaciones de vision artificial en la industria: http://www.jasvisio.com/aplicaciones-vision-artificial-industria.html

Lozano Nasner, M., & Bayona Ibañez, E. (2003). Sistema para la clasificación de objetos en un proceso industrial utilizando técnicas de visión artificial. Recuperado a partir de http://200.93.148.28/drupal/files/s9alyQ9U4IIZNHq.pdf.

Malpartida, E. S., & Sotelo, J. C. (s. f.). Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot.

Porras, J., De la Cruz, M., & Morán, A. (2014). Clasification system based on computer vision.

Saldaña González, G., Estévez Carrerón, J., & Silva Ortigoza, R. (2014). Sistema automático de clasificación aplicado a la industria farmacéutica. Recuperado a partir de

http://somi.ccadet.unam.mx/somi29/memoriassomi29/PDFS/Intrumentacion/120-GJSOMI-132-120.pdf.

Sandoval Niño, Z. L., & Prieto Ortiz, F. A. (2007). Caracterización de café cereza empleando técnicas de visión artificial. *Revista Facultad Nacional de Agronomía*, Recuperado a partir de http://www.revistas.unal.edu.co/index.php/refame/article/view/24461.

Sofu, M. M., Er, O., Kayacan, M. C., & Cetisli, B. (2016). Design of an automatic apple sorting system using machine vision. *Computers and Electronics in Agriculture, 127*, 395-405.

Vanderplas, J. (2011). Scikit Learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, pp. 2825-2830. Obtenido de scikit-learn.org: http://scikit-learn.org/stable/modules/neighbors.html

Young-Jin, C., Kisung, Y., & Wooram, C. (2016). Vision based detection of loosened bolts using Hough transform and support vector machines. *Automation in Construction, 71*(2), 181-188.

Zurita, J., & Pérez, L. (2014). Diseño e implementación de una máquina automática clasificadora de objetos según su color detectados mediante un sensor de color y clasificados por un brazo robótico. 1-8.